The PCP Theorem in

Real Number Complexity Theory

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

(Dr.rer.nat.)

genehmigte Dissertation

vorgelegt von

Master of Science (M.Sc.)

Willem Martijn Baartse

geboren am 13. Dezember 1985 in West-Voorne (Niederlande)

Gutachter: Prof. Dr. Andreas Goerdt

Gutachter: Prof. Dr. Klaus Meer

Gutachter: Prof. Dr. Luis Miguel Pardo

Tag der muendlichen Pruefung: 27. November 2015

Abstract

In this thesis we consider probabilistically checkable proofs (PCPs) in the model of computation working with real and complex numbers as introduced by Blum, Shub and Smale (the BSS model). Starting point of this thesis is the so-called PCP theorem, a major result in complexity theory. Roughly speaking this theorem says that one can check the correctness of certain proofs of mathematical statements with high probability by only looking at small parts of the proof. PCPs have a natural formulation in the BSS model as well. Considering the importance of the PCP theorem it seems natural to ask whether a similar theorem holds in this model also. The PCP theorem in the Turing model has two qualitatively very different proofs. The original proof by Arora et al. uses mostly algebraic techniques. Later another, combinatorial proof was discovered by Dinur.

In this thesis we give two corresponding proofs in the BSS model. Thereby we establish the PCP theorem in the framework of the real numbers as well as in the framework of the complex numbers. The main difficulties occur due to the fact that the structures $\mathbb{R}$ and $\mathbb{C}$ over which we work are uncountably infinite. For both classical proof techniques partially really new ideas are necessary to apply their structure to the new computational model. Apart from the importance of the PCP theorem it helps to better understand the similarities and differences between both models of computation. We hope that the new ideas we develop may be useful in other contexts as well.

While working on this thesis I was partly supported by the Deutschen Forschungsgemeinschaft with the projects ME 1424/7-1 and ME 1424/7-2. Hereby I gratefully acknowledge this support

Zusammenfassung

In dieser Arbeit betrachten wir probabilistisch prüfbare Beweise (Englisch: probabilistically checkable proofs PCPs) in einem Rechenmodell, das mit reellen und komplexen Zahlen arbeitet und 1989 durch Blum, Shub und Smale eingeführt wurde, dem BSS-Modell. Ausgangspunkt der Arbeit ist das sogenannte PCP-Theorem, eines der wichtigsten Resultate in der Komplexitätstheorie der letzten Jahrzehnte. Sehr grob gesprochen besagt es, dass man die Korrektheit von Beweisen mathematischer Aussagen mit hoher Wahrscheinlichkeit überprüfen kann, indem man nur sehr wenige Teile des Beweises untersucht. PCPs können auf natürliche Weise auch im BSS-Modell formuliert werden. Im Hinblick auf die Wichtigkeit des PCP-Satzes ist es daher naheliegend zu fragen, ob ein vergleichbares Ergebnis auch in diesem Modell gilt. Das klassische Theorem im Turingmodell hat zwei qualitativ sehr unterschiedliche Beweise. Der ursprüngliche Beweis von Arora et al. benutzt vorwiegend algebraische Techniken. Ein später gegebener Beweis von Dinur basiert im Wesentlichen auf kombinatorischen Methoden.

In der vorliegenden Arbeit präsentieren wir zwei entsprechende Beweise im BSS-Modell. Der PCP-Satz wird dabei sowohl im reellen als auch im komplexen Rahmen bewiesen. Die wesentlichen neuen Schwierigkeiten bei beiden Beweisen resultieren aus der Überabzählbarkeit der zugrundeliegenden Strukturen $\mathbb{R}$ und $\mathbb{C}$. Für beide klassischen Beweismethoden erfordert dies zum Teil wesentlich neue Ideen, um ihre Struktur auf das neue Rechenmodell zu übertragen. Neben der Bedeutung des PCP-Satzes an sich hilft dies, die Ähnlichkeiten und Unterschiede zwischen beiden Rechenmodellen besser zu verstehen. Es ist zu hoffen, dass die neuen Ideen, die wir hier entwickeln, auch in anderen Kontexten nützlich sind.

Während der Arbeit an dieser Dissertation wurde ich teilweise von der Deutschen Forschungsgemeinschaft im Rahmen der Projekte ME 1424/7-1 und ME 1424/7-2 gefördert. Für diese Unterstützung möchte ich mich hiermit bedanken.

Contents

Chapter 1

Introduction

Since at least the 1970's the Turing machine model has generally been accepted in theoretical computer science and algorithmic mathematics as the standard model for algorithms that work over strings and finite alphabets. This certainly still is the case. Nevertheless, in the last decades one can notice an increasing interest in studying alternative models that formalize algorithms working over other than finite structures as well. Research fields representing such approaches are quantum computing [38], bio-computing [27], neural networks [30], analogue computers and the analysis of dynamical systems as computational devices [16], recursive analysis [44] and, last but not least, the area of algebraic computability and complexity [17].[1] All these models in one or the other way focus on different aspects of computability and complexity by either working over different, often uncountable structures and/or allowing different operations than the Turing machine does.

Let us stress that we do not consider one such model as the only reasonable one, but all of them focus on certain aspects neglecting others. In order to estimate their value one has to carefully examine in which situations and for what questions the used model is appropriate. In this thesis we do not want to enter into a deeper discussion of that point.

The model which is central for all what follows herein was introduced in 1989 by Blum, Shub, and Smale in [15]. It is an algebraic model that formalizes algorithms over general structures with a focus on the real and complex numbers. In a certain way it homogenizes previous models in algebraic complexity theory by introducing a notion of uniformity. The basic characteristics of the model when dealing with $\mathbb{R}$ are the following: algorithms work with real numbers as entities; it is allowed to perform the basic arithmetic operations $+, -, \bullet, /$ as well as a branch operation reflecting the order: is $x \geq 0$? Complexity of algorithms is measured by counting the number of the above operations, and the size of a problem instance is (basically) the number of reals necessary to define it. A uniform algorithm then is a finite list of such operations working on an in principle countably infinite number of registers storing real numbers.

[1] The cited references are only thought as starting point for the more interested reader and by no means complete.

Other structures can be considered as well. For computations with the complex numbers test operations take the form: is $x = 0$? Also the Turing model is recaptured by considering computations over a two-element field.

The approach gives the possibility to introduce complexity classes over structures like $\mathbb{R}$ and $\mathbb{C}$ in a way completely analogous as it is done in the Turing model. Two of the main classes to be defined precisely below are $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$, the real analogues of the discrete classes reflecting polynomial time decidable and verifiable decision problems, respectively.

There are at least two motivations for studying such a model. One of them is the use of the computer for scientific computations. In a reasonable idealization of these kinds of computations the cost of multiplying two numbers can be viewed as independent of the size of the numbers (but see [45] for a more severe discussion in which situations the use of such a model is appropriate and when it is not). Moreover, any lower bound on the number of arithmetic operations necessary to solve a problem applies as well to algorithms working on the bit level.

Another motivation is to bring two well developed areas together, namely the theory of discrete computation on the one side and the theory of algorithmic analysis, algebra and topology on the other side. One line of investigations in the Blum-Shub-Smale model (henceforth abbreviated as BSS model) thus is to compare open problems, theorems and proof methods in the respective settings. This might clarify similarities and differences between both models and hopefully helps to give inside into the nature of computation both with bits and with real numbers. It reveals whether theorems, notions, ideas and arguments only apply specifically to the Turing/BSS model or have a broader validity. It as well leads to interesting new research questions. Lots of different results have been obtained in this respect, some of them establishing similarities, some of them differences. Here are a few examples.

The probably most interesting open question in both models is the famous P versus NP problem. As with the Turing model at the time of writing this thesis its answer is unknown both over the real and the complex numbers. There have, however, been some relations established showing that certain major open questions in the complexity theory of the respective models are closely related. One example is the P versus NP problem in the complex model and its connection to the relation of classes NP and BPP of discrete problems solvable by efficient probabilistic computations (result independently shown by Koiran and Smale). Another example is the interplay between the classical P versus NP problem and its variant in the so-called additive real number model [25, 26]. A first example of a classically valid theorem that holds as well in the real world by applying quite similar proof techniques is the existence of $\mathrm{NP}_{\mathbb{R}}$-complete problems shown in [15], see also below in Chapter 2. Another theorem which holds in both the Turing and the real BSS setting but this time for very different reasons is the decidability of all problems in $\mathrm{NP}_{\mathbb{R}}$ in single exponential time. For the Turing model this relies on a trivial counting argument. For the real (and complex) number model the class $\mathrm{NP}_{\mathbb{R}}$ is also decidable in single exponential time, but now simple counting arguments do not work. The search space for witnesses is not any longer finite but uncountable because certificates now become

strings of real numbers. It is thus a priori not even clear that all problems in $\mathrm{NP}_{\mathbb{R}}$ are decidable. The proof of this result has seen a long history. It is based on the theory of quantifier elimination algorithms, starting with Tarski's early work [43] and leading to single exponential time algorithms for the existential fragment only around the 1990's by quite involved techniques [28, 23, 40].

A theorem that holds over the reals by easier (though not trivial) arguments is Post's theorem about the existence of undecidable problems below the (real) Halting Problem [37]. Another theorem which needs considerable new techniques is a real variant of Toda's inclusion of the polynomial hierarchy in $P^{\#P}$ [12]. Ladner's theorem stating that unless P=NP there exist problems neither in P nor NP-complete provides an example where the more general point of view studying the problem in different structures sheds as well some light on why it holds classically and puts the original problem into a more model-theoretic framework [19]. It is currently known to hold over the complex numbers but open over the reals, see [6] for more information.

It should be stressed that of course the model also inspired work on problems that are not appropriately captured by the Turing approach. As one such we only mention the huge amount of deep work on homotopy methods to solve polynomial systems numerically. See [13] as a starting reference.

The main content of this thesis is to add another famous classical theorem to the list of results that hold as well in the BSS setting, namely the PCP theorem, where PCP stands for Probabilistically Checkable Proofs. The PCP theorem, having been proved in the early 1990's certainly is one of the milestones in discrete complexity theory within the last two decades. The theorem itself gives a highly surprising characterization of NP by (informally) stating that for verifying a mathematical proof probabilistically it is sufficient to inspect only a constant number of components. Being interesting by itself the theorem also has important applications in the field of approximation algorithms.

Let us first describe its historical environment and discuss on a higher informal level the two existing proofs of the classical PCP theorem. For this part see also the historical accounts [9, 10, 22]. Then, we shall outline the main contributions of the thesis illuminating the problems to face when proving a real and complex analogue of the theorem. All concepts touched briefly in the next lines will be precisely defined later on.

The usual interpretation of "efficiently verifiable" as represented by the definition of NP is checking efficiently a given potential proof of a statement. If a statement is false no proof should be accepted, if it is true there has to be a proof that is accepted. Using this notion, in general each part of the proof has to be inspected. But are there reasonable other notions how to verify a proof efficiently? This research was started in the mid eighties by several people. The idea was to consider an unlimited powerful prover trying to prove some assertion to a verifier which has to work in polynomial time. In this framework the NP notion corresponds to letting the prover write down a proof which the verifier has to check deterministically. Then it should decide whether it wants to accept or reject. With this idea in mind the NP notion may possibly seem a bit weak: Why should the verifier not be allowed to ask questions and to interact with the prover? Moreover, if the verifier

is allowed to accept incorrect assertions with small probability, then it might do better by using randomness in its verification process. Randomness could especially be useful in determining what questions to ask, so that the prover (despite its unlimited powerfulness) cannot know in advance which questions will be asked by the verifier. The complexity class IP (Interactive Proofs) is defined as the set of languages for which there exists a correctly working verifier as described above. Because randomness or interaction, when considered separately, do not add much power, most researchers suspected at first that IP would not be much bigger than NP. Therefore, it was a remarkable result when Shamir [42] proved that this notion is strong enough to capture polynomial space computations, i.e., the equality IP = PSPACE holds. Another possibility which can make the verifier even more powerful is interacting with even more provers which are not allowed to communicate with each other. The verifier can take advantage of this by forcing them to answer non-adaptively. This basically means that the best the two provers can do is to agree in advance on a strategy and fix their answers to every possible question the verifier may ask. An equivalent (and more common) way to view this setting is that the verifier makes queries to an oracle tape. The complexity class defined that way is denoted by MIP where 'M' stands for Multiple provers. Around the same time of Shamir's result it was shown in [34] that MIP = NEXP.

Researchers next tried to downscale this result to obtain an alternative characterization of NP. The approach to do this that turned out to be most successful is the introduction of PCP classes of which MIP is a special case: MIP = PCP(poly n, poly n)). The parameters in brackets denote the restrictions (up to a constant factor) on the amount of randomness and the number of queries, respectively, measured in dependence of the input size. Using this formalism, NP = PCP(0, poly n) because a natural NP verification algorithm uses no randomness and is allowed to inspect the entire proof. The famous PCP theorem states that NP = PCP(log n, 1) meaning that a PCP verifier for a problem in NP only needs a constant number of queries, independently of the input size!

As already mentioned the PCP theorem turned out to be very useful in obtaining lots of inapproximability results for combinatorial optimization problems. May be the one closest related to the PCP theorem is the non-existence of a polynomial time approximation scheme for the MAX-SAT problem [2]. We shall meet a similar approximation problem below.

Up to now there exist two qualitatively very different proofs of the PCP theorem. The original proof by Arora et al. [3, 2] combines a lot of algebraic techniques in a complicated way. Its basic ideas can be described as follows. Without loss of generality it is sufficient to construct a verifier for the NP-complete SAT problem respecting the required resource bounds. The SAT problem asks whether a given propositional formula in conjunctive normal form has a satisfying assignment. The natural NP-verification for this problem would take an assignment as proof that a formula is satisfied and then just plugs it into the clauses of the formula evaluating it. In order to obtain a uniform verification algorithm this obviously requires all components to be read by the verifier. Being only allowed to inspect a constant number of proof components the verifier must code satisfying assignments of a

formula in a completely different way. Towards this aim it uses different codes that allow for a stable error correction. The verifier now expects the oracle tape to contain such an encoding. It then has to perform two tasks. The first is to check that the proof certificate specified by the oracle tape is in some sense close to a codeword, i.e., the correct encoding of some assignment. The second step is a check; assuming the specified proof certificate is close to a codeword the checking procedure rejects with high probability if the encoded assignment is not satisfying. If the proof certificate is the correct encoding of a satisfying assignment the verifier will accept the proof.

Now two such PCP verifiers for NP are constructed in the proof by Arora et al. with different restrictions. One of the verifiers requires a polynomial amount of randomness and a constant number of queries. It uses the so-called Walsh-Hadamard code which replaces strings in $\{0,1\}^n$ by the truth table of the linear function generated by the string as coefficient vector. This makes the proof exponentially long. The verifier probabilistically checks whether a given truth table comes with high probability from a linear function and whether the underlying bit string is a satisfying assignment (actually, several linear functions are needed here). The second verifier uses a logarithmic amount of randomness and a polylogarithmic number of queries. In principle it uses similar ideas as the first, this time coding strings via low-degree polynomials in a finite field. The algebraic techniques necessary here are however much harder than in the linear case and constitute one of the cornerstones in the entire proof. These two verifiers finally are combined with a new technique called verifier composition to obtain a verifier for NP which shares the better parameters of the two verifiers already constructed. It uses a logarithmic amount of randomness and a constant amount of queries. For this technique to be applicable it is crucial that the second verifier can be put into a segmented form. This means that it treats the oracle tape as partitioned into a number of segments and it queries only a constant number of these segments. In a simplified view, the verifier obtained by composing the two others works just as the second verifier with the following difference. Instead of reading the segments completely it expects the segments to be encoded again. Then it uses the first verifier to query only a few bits in the encoding so that the composed verifier respects the necessary restrictions (logarithmic randomness, constant number of queries) to prove the PCP theorem.

The second proof of the classical PCP theorem was given in 2005 by I. Dinur [21]. It uses a very different approach based on combinatorial techniques which exploit the nice properties of expander graphs. Starting point is an extension of the 3-SAT problem, a so-called constraint satisfaction problem CSP. Instances of such a problem consist of several constraints each of which depends on a certain fixed number of variables. The variables can take values from some finite alphabet and the instance is called satisfiable if there exists an assignment satisfying all constraints. Instead of constructing complicated encodings for the proof certificate, Dinur constructs a so-called gap reduction for the input. A gap reduction reduces a CSP instance to another CSP instance such that satisfiability is maintained. The crucial point is what happens with unsatisfiable instances. A common many-one reduction as it is used for defining completeness only requires that unsatisfiability is maintained as

well implying that for each assignment at least one constraint of the reduced instance is not satisfied. Gap reductions require more, namely that unsatisfiable instances are mapped to instances having the property that for every assignment at least an $\epsilon > 0$ fraction of constraints is unsatisfied; of course, $\epsilon > 0$ should be independent of the input instance. It was known before (and actually is easy to see) that the PCP theorem is equivalent to the existence of such a gap reduction. Dinur constructs one such in a logarithmic number of rounds where in each round the gap is doubled. A round consists of a lot of rearranging and grouping of constraints and changing the domains of variables. Properties of expander graphs are used here to increase the gap. After increasing the gap in which the finite alphabets used as domains for variables increase in size, an alphabet reduction has to be performed and the alphabet is reduced again to one of constant size. Then the next round can start. To achieve such an alphabet reduction also Dinur's proof relies on a kind of verifier composition which uses the long transparent proof verifier from the proof by Arora et al., i.e. the one requiring polynomial randomness and a constant number of queries.

This was the starting point for our project and this thesis. The allover goal was to prove a real and complex version of the PCP theorem in the BSS setting. The definition of the corresponding concepts like real verifiers and PCP classes is almost straightforward and the question makes perfect sense in the new framework as well. In this thesis we prove real and complex analogues of the PCP theorem along the lines of both classical proofs.

At a first glance one might expect the proof by Arora et al. to fit better into the real number model because of its very algebraic nature. In the end, systems of polynomials and their solvability is the crucial $\mathrm{NP}_{\mathbb{R}}$-complete problem. In contrast, Dinur's proof relies a lot on different kinds of satisfiability problems over changing finite alphabets, graph theory and combinatorics and thus seemingly has a much more discrete flavour. It turns out that the opposite is true. At least with our current knowledge the ideas behind Dinur's proof can be transferred more smoothly (though not trivially) to the real and complex numbers. One important point to do so is to reconsider the typical $\mathrm{NP}_{\mathbb{R}}$-complete problem dealing with solvability of polynomial systems in such a way that the role of different finite alphabets in the classical proof is taken over by different sets of real variables occurring in polynomial systems. The alphabet size is then replaced by the cardinality of such variable sets, the latter always ranging over the reals as underlying domain. A second point is to attach a canonical graph to this new $\mathrm{NP}_{\mathbb{R}}$-complete problem so that the ideas about expanders can be used again. Then fundamental steps in our proof as well are the design of a gap-amplification for this new complete problem together with a dimension reduction. The former of course has to be done for a particular complete problem in the real model, the latter replaces the alphabet reduction in Dinur's proof. Not surprisingly, all this has to be adapted accordingly with a lot of details to be checked. But basically it turns out that a proof can be set up that follows very similar ideas as Dinur's. Moreover, the proof nowhere uses arguments relying on the ordering of the underlying field of reals, and so it also establishes the PCP theorem over $\mathbb{C}$.

In contrast, an algebraic proof of the real PCP theorem results in considerably more difficulties. As described above the proof by Arora et al. constructs a $(\log(n), \text{polylog}(n))$-restricted verifier using low-degree algebraic polynomials as coding objects for potential satisfying assignments. When trying the same in the real setting immediately significant differences arise. The first question is that about the domain of such polynomials. Classically, everything is embedded into finite fields of suitable cardinality, so any such polynomial can be represented via a finite function value table. Over the reals, formally the domain for a polynomial coding a potential zero of a given polynomial system of course is the entire space and thus not useful. So one must restrict it to a finite subset on which function values are given as (part of the) proof to the verifier. But it is unclear how this domain should look like. First, it has to depend on the coefficients of the input system and thus might change with each new input. Next, any choice which seems appropriate lacks the nice structure finite fields have. For example, for many tests performed on such an encoding one needs the sum and/or product of two elements in the domain to be again in the domain in order to compare the resulting values of the coding polynomial. This has several drawbacks: Domains that might work explode with respect to their cardinality, and many invariants of the uniform distribution exploited in the classical proof over finite fields are lost. Note that similar problems already occur in the design of a long transparent proof for $\text{NP}_{\mathbb{R}}$ using linear functions as coding objects [35, 5]; there, however, they are harmless because of the way long transparent proofs are used later on in the proof of the full $\text{PCP}_{\mathbb{R}}$ theorem, see below. A first idea to circumvent these problems is to still take advantage of using finite fields by choosing as coding objects certain trigonometric polynomials which have as domains finite fields but map the latter to the reals. Though this gives a certain nice structure back it creates serious new problems. Typical tests which aim to verify that a function value table represents a coding object like a polynomial make use of univariate restrictions. The idea is that if a given multivariate function restricted to certain one-dimensional curves corresponds to a univariate function of the appropriate type, then this also holds for the original multivariate function. So for many tests the verifier additionally requires to get such univariate restrictions from the prover; it then compares them with the given function in order to verify it to be a low-degree polynomial. But univariate restrictions of trigonometric polynomials behave quite different compared to those of algebraic polynomials. For example, the degree of a restriction along a line can get much larger. This creates severe problems with respect to questions like proof sizes. It results in a lot of necessary pull-ups in order to specify a good subset of univariate restrictions sufficient to characterize multivariate trigonometric polynomials. Another consequence of this modified low-degree test will be that the test does not have an appropriate format to be used further on. The classical technique of verifier compositions heavily relies on the structure of the algebraic low-degree verifier in order to compose it with other verifiers. The decisive criterion here is segmentation. It means that even if a polylogarithmic number of proof components are inspected by the verifier, they are inspected in a very structured form, namely reading constantly many proof segments of polylogarithmic size. This will be the case with the trigonometric low-degree test we

construct. So in an additional step we can use it to put a non-segmented verifier into a segmented form, a requirement needing significant new ideas as well. Only then we are able to compose in the final step the resulting verifiers to get once again the full $\mathrm{PCP}_{\mathbb{R}}$ theorem, this time by using an algebraic approach.

The thesis is structured as follows. In Chapter 2 we introduce the basic notions and complexity classes. This includes a short description of the Blum-Shub-Smale model over $\mathbb{R}$ and $\mathbb{C}$, its main complexity classes important for this work as well as defining the concept of probabilistically checkable proofs in this framework. Chapter 3 recalls the construction of long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ from [35, 6]. Though it is not a central part of the PhD thesis and was known before, for sake of completeness we decided to include the proof. This is necessary for a full understanding of the thesis' main parts, i.e., how long transparent proofs enter the two proofs of the real PCP theorem. The reader might skip this part at a first reading. The main contribution of the thesis are Chapters 4, 5 and 6. In Chapter 4 a low-degree test for trigonometric polynomials is developed which is of great importance in the proof of the real PCP theorem in Chapter 5 along the lines of Arora et al. Chapter 6 gives an alternative and less complicated proof of the real PCP theorem along the lines of the proof by Dinur. Both proofs take their starting point in the classical approach as described above and then considerably deviate due to the infinite underlying structure. We attempt at all places to point out to the reader where the main differences to the discrete proofs lie. The text, however, is written in a self-contained fashion for readers not necessarily knowing the classical proofs. We decided to arrange the chapters according to the time order in which the corresponding classical proofs were achieved. Actually, in the BSS framework the Dinur-like proof was the first that was shown. The reader can decide him/herself which proof to read first. Both can be studied independently of each other.

The thesis ends with a short summary and a discussion of open interesting research topics around the results presented herein.

The results of this thesis have been partially published already in [5, 6, 7]. We closely follow the proofs as presented in those papers.

Chapter 2

Basic notions and complexity classes

In this chapter we present the basic definitions, notions and concepts that will be needed throughout the rest of the thesis. The two central notions in this thesis are BSS machines and probabilistically checkable proofs. Readers interested in a more thorough introduction into BSS theory are referred to [14, 15]. A good introduction into the theory of PCPs can be found in [1].

Below we will mainly be dealing with BSS machines over $\mathbb{R}$. In the proofs in this thesis we do not use the division operation. We will often talk about BSS machines and implicitly assume that they work over $\mathbb{R}$. Most things work the same for BSS machines over $\mathbb{C}$. In cases where things are different we will shortly point this out. We define complexity classes $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$ and we will define PCP classes over $\mathbb{R}$. With these definitions at hand we can then make precise the results that we will derive in this thesis.

2.1 Definition of a BSS machine over $\mathbb{R}$

As described in the introduction, a BSS machine is basically a Turing machine which manipulates real numbers as basic entities and can perform the arithmetic operations $+,-,\bullet,/$ and the test $x \geq 0$ at unit cost. A BSS machine over $\mathbb{C}$ has the test $x = 0$ instead of $x \geq 0$ because there is no order on $\mathbb{C}$. In this section we will give a more formal definition. BSS machines were first defined in [15].

An algorithm in the Turing model works over some finite alphabet Σ and at each step there is some element of Σ^* on the work tape. Let us start by defining the equivalent of this for BSS machines over $\mathbb{R}$.

Definition 2.1. *Let $\mathbb{R}^\infty$ be the set of sequences of real numbers with finitely many non-zero components, i.e.,*

$$\mathbb{R}^\infty := \{(x_1, x_2, \dots) | x_i \in \mathbb{R} \text{ and } \exists k \in \mathbb{N} \forall j > k \, (x_j = 0)\}.$$

We proceed with a precise definition of how a BSS algorithm manipulates elements in $\mathbb{R}^\infty$. If one would replace $\mathbb{R}^\infty$ by $\{0,1\}^*$ in the definition below, then one would obtain something which is equivalent to a Turing machine.

Definition 2.2. *A BSS machine M over $\mathbb{R}$ is given by a finite set I of instructions labeled by $1,\ldots,N$. A configuration of M is a quadruple $(n,i,j,x) \in I \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^\infty$. The n indicates which instruction is currently being executed, i and j are used as addresses to perform copy instructions and x is the content that is contained in the registers of M. The length of x is defined as the largest index of a register whose content is not zero. The initial configuration of M on input x is $(1,1,1,x)$. If in the configuration of M the instruction number $n = N$ then the computation halts and the output is the content contained in the registers at that moment. The instructions of M can be of the following three types.*

a) computation: to a computation instruction n corresponds an integer s that is the address of the register whose content is changed in this computation. The content of the other registers remains unchanged, except for the copy registers. There are two possibilities. Either x_s is set to some constant α or x_s is set to the value $x_k \circ_n x_l$ where $\circ_n \in \{+,-,*,:\}$ and k and l are constants which depend only on n. The next instruction is $n+1$ and the value of the copy-register i is increased by 1 or set to 1. Similar for the copy-register j.

b) branch: in a branch node n, the next instruction depends on the value of x_1. If $x_1 \geq 0$, then the next instruction will be $\beta(n)$. Otherwise the next instruction is $n+1$. The content of all registers remains unchanged.

c) copy: The content of the register x_j is copied into the register x_i. So only the content of register i changes. The next instruction is $n+1$.

All α that appear in the computation instructions are called machine constants. Obviously, a machine M can have only a finite number of machine constants.

Convention 2.3. *a) Since there is no blank symbol to mark the end of the input we will assume that x_1 is an integer indicating this length.*
b) We also suppose that division by 0 does not occur. This can easily be implemented by testing whether the divisor is 0 before executing the division and if it turns out to be 0, then enter into an infinite loop.

Remark 2.4. *The copy-registers and -instruction are necessary because the computation instructions work on fixed registers. Without copying, the machine would only be able to access a fixed finite set of registers. That would not suffice to deal with arbitrary long inputs from $\mathbb{R}^\infty$. The way this copying is implemented may seem rather restrictive because no indirect addressing is possible. However, it is general enough for our purposes.*

Now that we have defined what a BSS machine is, we are ready to define what it means for a machine to compute a function. This is very similar to the Turing case.

Definition 2.5. *Let M be a BSS machine.*

a) The set $\Omega_M := \{x \in \mathbb{R}^\infty | M \text{ halts on } x\}$ is the halting set of M. It is also called the language accepted by M.

b) The partial function $\phi_M : \Omega_M \to \mathbb{R}^\infty$ which is obtained in the obvious manner by performing the computation of M on an input x and taking the output as the result, is called the function computed by M.

c) A partial function $f : A \to \mathbb{R}^\infty$ is computable if there exists a BSS machine M which computes it.

d) A set $A \subseteq \mathbb{R}^\infty$ is called decidable if there exists a machine M computing the characteristic function of A. We also say that M decides A.

e) If for a function $f : \mathbb{N} \to \mathbb{N}$, a machine M always holds within at most $f(n)$ steps on an input of length n we say that M works in time $f(n)$. If a machine M working in time $f(n)$ decides a set $A \subseteq \mathbb{R}^\infty$, then we say that A is decidable in time $f(n)$.

2.2 The complexity classes $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$

With the above definition we are ready to define the complexity classes $\mathrm{P}_{\mathbb{R}}$, $\mathrm{NP}_{\mathbb{R}}$. These definitions are again very similar to the classical ones.

Definition 2.6. *($\mathrm{P}_{\mathbb{R}}$): The complexity class $\mathrm{P}_{\mathbb{R}}$ is defined as the set of languages $A \subseteq \mathbb{R}^\infty$ for which there exists a polynomial function $f(n)$ and a machine M deciding A in time $f(n)$. We also say that A is decidable in polynomial time.*

Definition 2.7. *($\mathrm{NP}_{\mathbb{R}}$): The complexity class $\mathrm{NP}_{\mathbb{R}}$ is defined as the set of languages $A \subseteq \mathbb{R}^\infty$ for which there exists a machine M and a polynomial function $f(n)$ such that:*

a) for every $x \in A$ of length n there exists a $y \in \mathbb{R}^\infty$ such that M accepts (x, y) within at most $f(n)$ steps

b) for every $x \notin A$ of length n it holds for all $y \in \mathbb{R}^\infty$ that M rejects (x, y) within at most $f(n)$ steps.

Here we can define (x, y) for example as $(x_1, x_2, \ldots, x_{x_1}, y_1, y_2, \ldots)$ because we assumed that x_1 indicates the length of x.

This defenition can be interpreted as follows. A verifier gets an input and wants to decide whether this input belongs to the language A. A prover tries to convince the verifier that this is indeed the case. If the input belongs to A, then the prover should be able to convince the verifier by giving a correct proof of this fact. If the input does not belong

to A, then the prover should not be able to trick the verifier into thinking that the input does belong to A.

The definitions of $\mathrm{P}_{\mathbb{C}}$ and $\mathrm{NP}_{\mathbb{C}}$ are analogous. From this definition it is obvious that proving decidability of $\mathrm{NP}_{\mathbb{R}}$ is not as easy as proving the decidability of NP. We cannot decide $\mathrm{NP}_{\mathbb{R}}$ by simply trying all possible certificates since there are uncountably many. As mentioned in the introduction, there do exist clever quantifier elimination algorithms [28, 23, 40] that show that despite the uncountable size of the search space all problems in $\mathrm{NP}_{\mathbb{R}}$ are decidable in single exponential time just as in the Turing model. So we have

Theorem 2.8. *All problems in* $\mathrm{NP}_{\mathbb{R}}$ *and* $\mathrm{NP}_{\mathbb{C}}$ *are decidable in single exponential time, in the respective BSS model.*

The next obvious question is whether there exist (natural) $\mathrm{NP}_{\mathbb{R}}$-complete problems. So let us first define a notion of reducibility and let us then define what it means for a problem $A \subseteq \mathbb{R}^\infty$ to be $\mathrm{NP}_{\mathbb{R}}$-complete. This is just what one would expect and nothing surprising happens at this point.

Definition 2.9 (reducibility)**.** *A problem $B \subseteq \mathbb{R}^\infty$ is reducible to a problem $A \subseteq \mathbb{R}^\infty$ if there exists a function f which can be computed in polynomial time by a BSS machine and satisfies $x \in B \Leftrightarrow f(x) \in A$*

Definition 2.10 ($\mathrm{NP}_{\mathbb{R}}$-completeness)**.** *A problem $A \in \mathrm{NP}_{\mathbb{R}}$ is $\mathrm{NP}_{\mathbb{R}}$-complete if every problem $B \in \mathrm{NP}_{\mathbb{R}}$ is reducible to A.*

Again the situation is similar to the situation in the Turing model and in the BSS model there exist natural $\mathrm{NP}_{\mathbb{R}}$-complete problems, though the number of known natural $\mathrm{NP}_{\mathbb{R}}$-complete problems is much smaller than the number of known natural NP-complete problems. The following problem turns out to be $\mathrm{NP}_{\mathbb{R}}$-complete.

Definition 2.11 (Hilbert Nullstellensatz)**.** *The Hilbert Nullstellensatz over $\mathbb{R}$ is the problem of deciding whether a set $\mathcal{P} = \{p_1, \ldots, p_m\}$ of multivariate polynomials has a common zero.*

The $\mathrm{NP}_{\mathbb{R}}$-completeness of the Hilbert Nullstellensatz problem over $\mathbb{R}$ is shown in [15] using ideas from Cook's theorem that SAT is NP-complete

Theorem 2.12 (Blum, Shub, Smale)**.** *The Hilbert Nullstellensatz problem over $\mathbb{R}$ is $\mathrm{NP}_{\mathbb{R}}$-complete and the Hilbert Nullstellensatz problem over $\mathbb{C}$ is $\mathrm{NP}_{\mathbb{C}}$-complete.*

Just as in the Turing model the problem SAT can be reduced to 3SAT, the Hilbert Nullstellensatz problem can be reduced to a further restricted problem which we define below.

Definition 2.13 (Quadratic Polynomial System (QPS))**.** *An instance of a QPS problem is a set of $\mathcal{P} = \{p_1, \ldots, p_m\}$ of multivariate quadratic polynomials and every quadratic polynomial $p \in \mathcal{P}$ must be of the form $x_i - (x_j \circ x_k)$ or $x_i - c$ where $\circ \in \{+, -, *\}$ and $c \in \mathbb{R}$. $j = k$, $i = j$ and $i = k$ are allowed. The set $\mathcal{P} = \{p_1, \ldots, p_m\}$ is a yes-instance if the set of polynomials has a common zero.*

It is not difficult to show that the Hilbert Nullstellen problem is reducible to the QPS problem. This is a standard lemma in BSS theory. We prove it here because the proof is relatively simple and we will need it several times later on. In the proof the notion of a straight-line program is used. A straight-line program is analog to a boolean circuit. Instead of 0 and 1 real numbers are being used and the gates can perform the basic arithmetic operations. For details see [17]

Lemma 2.14. *There exists a function f which can be computed in polynomial time on a BSS machine which has the property that if $\mathcal{P}$ is an instance of the Hilbert Nullstellen problem, then $f(\mathcal{P})$ is an instance of the QPS problem and $\mathcal{P}$ has a common zero if and only if $f(\mathcal{P})$ has a common zero*

Proof. From $\mathcal{P}$ a straight-line program without divisions and scalar multiplications can be constructed which computes all polynomials in $\mathcal{P}$. This can be done in polynomial time. Let us now construct the set of quadratic polynomials $f(\mathcal{P})$ from this straight-line program. We add all equations of the straight-line program to $f(\mathcal{P})$ and for all variables y_i corresponding to a polynomial in $\mathcal{P}$ we add the equation $y_i - 0 = 0$ to $f(\mathcal{P})$.

Consider for example $\mathcal{P} = \{x_1x_3 - x_2 + 5, x_3^2 + x_1^2\}$. A straight-line program computing both polynomials could look like this:

$$\begin{aligned} y_1 &= x_1x_3 \\ y_2 &= y_1 - x_2 \\ y_3 &= y_2 + 5 \\ y_4 &= x_3^2 \\ y_5 &= x_1^2 \\ y_6 &= y_4 + y_5. \end{aligned}$$

For this straight-line program we have

$$f(\mathcal{P}) = \{y_1 - x_1x_3, y_2 - (y_1 - x_2), y_3 - (y_2 + 5), y_4 - x_3^2, y_5 - x_1^2, y_6 - (y_4 + y_5), y_3 - 0, y_6 - 0\}.$$

It is clear that $\mathcal{P}$ has a common zero if and only if $f(\mathcal{P})$ has a common zero. □

There are more important and interesting complexity classes in the BSS model. In this thesis the classes presented here ($\mathrm{P}_{\mathbb{R}}$, $\mathrm{NP}_{\mathbb{R}}$ and $\mathrm{PCP}_{\mathbb{R}}$ (next subsection)) are most important.

2.3 PCP classes in the BSS model

Now that we have taken a look at the basic complexity classes in the BSS model let us define the complexity classes that are the central topic of this thesis.

Basically, the PCP classes are randomized versions of the non-deterministic complexity classes with a restriction on the number of components that the verifier is allowed to query.

Before we get to the definition of the $\mathrm{PCP}_{\mathbb{R}}$ classes we first define how such a restricted randomized verification algorithm (also called a verifier) works.

Definition 2.15. *An $(r(n), q(n))$-restricted verifier is a BSS machine which works in polynomial time and has oracle access to a proof string π. This oracle access could for example be implemented as a different kind of copy instruction which copies the content of the jth register of π into the register x_i (where i, j is the content of the copy-registers of the machine). The verifier is allowed to use $O(r(n))$ random bits and may make $O(q(n))$ queries to π. Its output is either 'accept' or 'reject'.*

Note that the verifier uses random bits and not random reals. This is natural because the randomness is used to decide which registers in π it wants to query and that is a discrete decision. Note also that this essentially limits the length of π to $q(n) \cdot 2^{O(r(n))}$ and we would not have such an inherent limitation in the case of real randomness.

This notion of a verification algorithm now naturally leads to the definition of the $\mathrm{PCP}_{\mathbb{R}}$ classes, just as it is the case in the Turing model.

Definition 2.16. *For two functions $r, q, \mathbb{N} \to \mathbb{N}$ the complexity class $\mathrm{PCP}_{\mathbb{R}}(r(n), q(n))$ is defined as the set of languages A for which there exists a verifier V such that:*

a) for every $x \in A$, V accepts with probability 1

b) for every $x \notin A$, V rejects with probability at least $\frac{1}{2}$.

The definition of PCP classes over $\mathbb{C}$ is completely analogous.

To get a feeling for these $\mathrm{PCP}_{\mathbb{R}}$ classes let us look at a few examples. The class co-$\mathrm{RP}_{\mathbb{R}}$ in item b) is defined analogously to the class co-RP.

Example 2.17. *a)* $\mathrm{PCP}_{\mathbb{R}}(\log n, 0) = \mathrm{P}_{\mathbb{R}}$

b) $\mathrm{PCP}_{\mathbb{R}}(\mathrm{poly}(n), 0) = \text{co-}\mathrm{RP}_{\mathbb{R}}$

c) $\mathrm{PCP}_{\mathbb{R}}(\log n, \mathrm{poly}(n)) = \mathrm{NP}_{\mathbb{R}}$

d) $\mathrm{PCP}_{\mathbb{R}}(\log n, 1) \subseteq \mathrm{NP}_{\mathbb{R}}$

e) Consider the problem of deciding whether a QPS instance has a solution, given that we know that one of the following two alternatives hold.

- *All polynomials have a common zero.*
- *Any subset of the polynomials in the QPS problem which contains at least 90% of the polynomials does not have a common zero.*

This problem belongs to belongs to $\mathrm{PCP}_{\mathbb{R}}(\log n, 1)$.

a) holds because the behaviour of the verifier can be simulated for all possible random strings in polynomial time. So a polynomial time algorithm can check whether the verifier accepts for every possible random string of length $O(\log n)$. b) holds by definition. c) holds for the same reason as a). d) follows from c). To see that e) holds is less trivial and this is actually used in the Dinur proof. Consider the following verification procedure.

The verifier expects the proof string to specify an assignment for the variables occurring in the quadratic polynomials of the QPS instance. Since the verifier is allowed to make only a constant number of queries it cannot read the complete assignment. It can only choose constantly many variables and read the values assigned to those variables by the proof string. We show that this is sufficient. The verifier chooses the components it wants to read in the proof string in the following way. It selects at random a large enough (but constant) number of polynomials in the instance and it queries the values which the proof string specifies for the variables occurring in these polynomials. Then, it uses these values to evaluate these polynomials. If they all evaluate to zero, then the verifier accepts. Otherwise it rejects. Let us now prove that this verifier works correctly. Let us first assume that the instance has a common zero. In this case the proof string can specify this common zero and the verifier will accept no matter which polynomials it selects for evaluation. So the verifier works correctly in this case. Let us now assume that the instance does not have a common zero. In this case, no matter which assignment is specified by the proof string, there is at least a $\frac{1}{10}$ fraction of polynomials in the instance which do not evaluate to zero under the assignment specified by the proof string. Hence, with high probability the verifier selects one of these polynomials which do not evaluate to zero under the assignment specified by the proof string. Therefore, the verifier rejects with high probability.

This thesis is all about showing that the reverse inclusion of d) holds as well.

Theorem 2.18. $\mathrm{PCP}_{\mathbb{R}}(\log n, 1) = \mathrm{NP}_{\mathbb{R}}$ *and* $\mathrm{PCP}_{\mathbb{C}}(\log n, 1) = \mathrm{NP}_{\mathbb{C}}$

2.4 Expander graphs

In this section we collect a few results on expander graphs that we will need in Chapters 4 and 6.

Expanders are regular graphs that in a certain sense exhibit properties of random regular graphs of the same degree of regularity, see [24]. Expanders can be defined using two notions, algebraic and combinatorial expansion. For the definition of algebraic expansion we need the random walk matrix of an undirected multigraph G which can contain loops. This matrix describes the probabilities for moving from one vertex in the graph to one of its neighbors assuming a uniform distribution.

Definition 2.19. *(Random walk matrix) Let $G = (V, E)$ be a graph. The random walk matrix $A(G)$ of G is defined to be the $|V| \times |V|$ matrix in which the entry A_{ij} equals the probability that in a random walk on G vertex j is chosen after vertex i. Here each edge incident with node i and not being a loop is chosen with the same probability, whereas loops are chosen with twice this probability.*

Definition 2.20. *(Algebraic expansion) Let $n, d \in \mathbb{N}, \lambda < 1$, and $G = (V, E)$ a d-regular graph with $|V| = n$, i.e., every vertex has degree d. Let $\lambda(G)$ be the second largest eigenvalue in absolute value of $A(G)$. The graph G is called a d-regular expander with expansion parameter λ if $\lambda(G) \leq \lambda$.*

Another way of dealing with expanders is the following.

Definition 2.21. *(Combinatorial expansion) Let $n, d \in \mathbb{N}, \rho > 0$, and $G = (V, E)$ a d-regular graph with $|V| = n$. The graph G is called a d-regular edge expander with expansion rate ρ if for every subset $S \subset V$ with $|S| \leq \frac{n}{2}$ the number of edges between S and its complement $\bar{S} = V \setminus S$ is at least $\rho d|S|$.*

We list several properties relating these two notions that we will use at different places in the proofs. Proofs of all cited statements about expanders can be found, for example, in [1].

Theorem 2.22. *Let $d \in \mathbb{N}$ and let $G = (V, E)$ be a d-regular expander graph with expansion parameter $0 < \lambda < 1$. Then the following is true:*

a) For a set $S \subset V$ and its complement $\bar{S}$ the set $E(S, \bar{S})$ of edges between S and $\bar{S}$ satisfies

$$|E(S, \bar{S})| \;\geq\; (1 - \lambda) \cdot \frac{d \cdot |S| \cdot |\bar{S}|}{|V|};$$

b) G is a d-regular edge expander with expansion rate $(1 - \lambda)/2$.

The following well known fact about existence of certain explicit expander families is used for some reductions in Chapter 6 to control the fraction of unsatisfied constraints in certain QPS-instances. It has originally been shown in [39], a proof can also be found in [1], Theorem 21.19.

Theorem 2.23. *Let $0 < \lambda < 1$, then there exists an integer $d := d(\lambda)$ such that for all $n \in \mathbb{N}$ there exists a d-regular expander on n vertices having expansion parameter λ. Moreover, given n the corresponding expander graph can be constructed in polynomial time in n.*

Note that by Theorem 2.22 we obtain a similar statement for edge-expander graphs of a given expansion rate $< \frac{1}{2}$.

Finally, we also need the following lemma about sub-additivity of $\lambda(A)$ on stochastic matrices, here formulated via random walk matrices of regular graphs, see [1], Exercise 21.7. on page 457.

Lemma 2.24. *Let $G_1 = (V_1, E_1)$ be a d_1-regular and $G_2 = (V_2, E_2)$ be a d_2-regular graph on the same vertex set $V := V_1 = V_2$. Then $\lambda(G_1 \oplus G_2) \leq \frac{d_1}{d_1+d_2}\lambda(G_1) + \frac{d_2}{d_1+d_2}\lambda(G_2)$. Here, the sum $\oplus$ of the two graphs G_1 and G_2 is the graph $G := (V, E_1 \sqcup E_2)$.*

Chapter 3

Long Transparent Proofs

In this chapter we construct long transparent proofs in the BSS setting. The original result was first published by Meer in [35]. The description presented here follows very closely the one in [6]. We include it for sake of completeness because long transparent proofs are an essential ingredient in both the Dinur proof and the Arora et al. proof. This also holds for their real and complex counterparts. In those proofs some properties of these long transparent proofs will be important. Readers interested in the details can find them in this chapter.

Before we start with the construction of such long transparent proofs it is important to make a remark about the running time of a verification algorithm in relation to how these long transparent proofs enter into both proofs of the $\mathrm{PCP}_{\mathbb{R}}$ theorem.

Remark 3.1. *Classically, long transparent proofs are proofs of exponential size that can be checked by making a constant number of queries. In our setting the size of the proofs will be doubly exponential because it seems that they require certain functions to be specified on such large domains due to $\mathbb{R}$ or $\mathbb{C}$ as underlying structure.*

This means that a verification algorithm verifying such a proof needs to generate exponentially many random bits in order to make a random query. If we count this as time used by the verifier, then the whole verification procedure would be useless because by Theorem 2.8 the verifier could as well compute directly whether the input instance has a solution or not.

The point here is not so much the verification that the input instance has a solution. What will turn out to be important is that the verifier checks with a constant number of queries that the proof is close to a correct encoding of a solution of the input instance. These transparent long proofs will only be applied to instances of such small size that the double exponential size of the proof will still be small enough.

Strategy for constructing long transparent proofs

In this section we will discuss the existence of long transparent proofs for problems in $\mathrm{NP}_{\mathbb{R}}$ in detail. Here again things are usually the same over $\mathbb{C}$ and we will point it out when there are differences.

We shall construct such a verifier for the $\mathrm{NP}_{\mathbb{R}}$-complete problem QPS. The verifier will receive the following three objects as input: An instance of the QPS problem, a (possibly incorrect) proof of the existence of an assignment under which the polynomials evaluate to zero, and a sequence of random bits. The verifier outputs either "accept" if it believes the proof to be correct or "reject" otherwise. The polynomials and the proof will be in the form of a sequence of real numbers whereas the random string is a sequence over $\{0,1\}$. Randomness is used to decide which locations in the proof to query. Since the corresponding addresses can be coded discretely only discrete randomness is needed.

Throughout this subsection let n denote the number of variables of the input polynomials. Let $\mathcal{P} := \{p_1, \ldots, p_m\}$ denote the system. All polynomials p_i are of degree at most two and depend on at most three variables. For $r \in \{0,1\}^m$ define $P(x,r) := \sum_{i=1}^{m} p_i(x) \cdot r_i$. The following is easy to see: Let $x \in \mathbb{R}^n$ be fixed. If x is a common zero of all $p_i(x)$, then $P(x,r) = 0$ for all r. And if x is no common zero the probability for uniformly taken r that $P(x,r) = 0$ is at most $\frac{1}{2}$. We work with $P(x,r)$ in order to capture both the real and the complex case in common.

Of course, if we want to verify whether an $a \in \mathbb{R}^n$ solves the system it does not make sense to plug it into $P(a,r)$ and evaluate because this requires again reading all components of a. We therefore rewrite $P(a,r)$ as follows:

$$P(a,r) = E(r) + A \circ L_A(r) + B \circ L_B(r), \tag{3.2}$$

where functions E, A, B, L_A, and L_B have the following properties. A and B are linear functions with n and n^2 many inputs, respectively. The coefficient vectors that represent these mappings depend on the chosen a only. More precisely,

$$A : \mathbb{R}^n \mapsto \mathbb{R} \text{ such that } A(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i \cdot x_i \ \forall\ x \in \mathbb{R}^n;$$
$$B : \mathbb{R}^{n^2} \mapsto \mathbb{R} \text{ such that } B(y_{11}, \ldots, y_{nn}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \cdot a_j \cdot y_{ij} \ \forall\ y \in \mathbb{R}^{n^2}$$

The functions E, L_A and L_B are linear as well. They take as arguments inputs from $\mathbb{Z}_2^m := \{0,1\}^m$ and give results in the spaces $\mathbb{R}, \mathbb{R}^n$ and $\mathbb{R}^{n^2}$, respectively. It is important to note that these mappings do only depend on the coefficients of the polynomials $p_1, \ldots, p_m$ but not on a. Therefore, given the system and a random vector $r \in \mathbb{Z}_2^m$ these functions can be evaluated deterministically without inspecting a component of the verification proof. As an immediate consequence of equation (3.2), for evaluating $P(a,r)$ it is sufficient to know two function values of certain linear functions, namely the value of A in $L_A(r) \in \mathbb{R}^n$

and that of B in $L_B(r)$. The verifier expects from the verification proof to contain these two real values.

More precisely, the proof is expected to contain so-called linear function encodings of the coefficient vectors defining A and B. This means that instead of expecting the proof to just write down those vectors we do the following. We define a finite subset $\mathfrak{D}$ of $\mathbb{R}^n$ and require the proof to contain all values of $A(x) := a^t \cdot x$ for all $x \in \mathfrak{D}$; similarly for B and a subset of $\mathbb{R}^{n^2}$. In order to work out this idea several problems have to be handled. First, though A in principle is a linear function over all $\mathbb{R}^n$ the verification proof must be finite. It can only contain finitely many components representing values of A. Among these components we of course must find those values in arguments that arise as images $L_A(r)$ for $r \in \mathbb{Z}_2^m$. Secondly, the verifier cannot trust the proof to represent a linear function which maps $\mathfrak{D}$ to $\mathbb{R}$. All it can do is to interpret the proof as giving just a function A from $\mathfrak{D}$ to $\mathbb{R}$ and try to find out if it is linear. Thirdly, even if the functions A and B are indeed linear on their corresponding domains and encode coefficient vectors a and b the verifier has to find out whether b is consistent with a, i.e., whether the coefficient vector $\{b_{ij}\}$ defining B satisfies $b_{ij} = a_i \cdot a_j$.

To verify all requirements within the necessary resources and error bounds the verifier tries to realize the following tasks: It expects the proof to provide two function value tables representing A and B on suitable domains (to be specified). Then first it checks whether both tables with high probability represent a linear function on the respective domains and if 'yes' how to compute the correct values of those functions in a given argument with high probability. In a second part the verifier checks consistency of the two involved coefficient vectors with high probability. Finally, it evaluates (3.2) to check whether the result equals 0.

A correct proof will provide the tables of two linear functions on the appropriate domains of form $A(x) = a^t \cdot x$ and $B(x) = b^t \cdot x$ with vectors $a \in \mathbb{R}^n, b \in \mathbb{R}^{n^2}$ such that $b_{ij} = a_i \cdot a_j, 1 \leq i, j \leq n$. In this ideal case, equation (3.2) can be evaluated by reading only two components of the entire proof, namely one value of A and one of B. If the proof is correct the verifier will always accept.

Suppose then that the given QPS instance has no solution. The verifier has to detect this for any proof with high probability. There are different cases to consider where in the proof errors can occur. The first such case is the one in which one of the two functions which the proof provides is in a certain sense far from being linear. The verifier will be able to detect this with high probability by making only a few queries into the function value table and then reject. A more difficult situation occurs when the given function is not linear but close to linear. In this case the verifier's information about the proof is not sufficient to conclude that it is not completely correct. To get around this problem a procedure that aims to self-correct the values which the proof gives is invoked. For A and B as given in the tables we shall define self-corrections f_A, f_B. Assuming that the function value tables are almost linear will guarantee that these self-corrected functions are linear on the part of the domain which is important for us. Furthermore, the values of these self-corrected functions can be computed correctly with high probability at *any*

argument in this part of the domain by making use of constantly many other values in the table only. In case A is linear f_A equals A on the domain on which it is defined.

We will now carry out the following plan:

a) Define the domains on which we want the verification proof to define functions A and B;

b) check linearity of these functions such that if they are far from linear it will be discovered with high probability;

c) assuming no contradiction to linearity has been detected so far define the self-corrections f_A and f_B; use these to detect with high probability an error if consistency between the coefficient vectors of the two linear functions is violated;

d) for random $r \in \mathbb{Z}_2^m$ obtain the correct values of $f_A(L_A(r))$ and $f_B(L_B(r))$ with high probability and use these values together with $E(r)$ to evaluate $P(a, r)$. Check whether the result is zero.

3.1 Appropriate domains for linearity

We will now describe the domain $\mathfrak{D}$ on which the values of A should be provided by the proof. The domain on which we want the proof to define the function B will be constructed analogously.

The function $L_A : \mathbb{Z}_2^m \to \mathbb{R}^n$ which generates the arguments in which A potentially has to be evaluated has a simple structure depending on the input coefficients of the polynomials p_i. Written as a matrix its entries are either 0 or such coefficients, i.e., real numbers that constitute the QPS instance. Let $\Lambda := \{\lambda_1, \ldots, \lambda_K\}$ denote this set of entries in L_A, considered as a multiset. Since each p_i depends on at most 3 variables it is $K = O(m)$. In order to simplify some of the calculations below we assume without loss of generality that $m = O(n)$; if not we can add a polynomial number of dummy variables to the initial instance. Thus $K = O(n)$. Without loss of generality we also assume $\lambda_1 = 1$. The components of any vector occurring as argument of A now are 0-1 linear combinations of elements in Λ. We therefore define

$$\mathcal{X}_0 := \{\sum_{i=1}^{K} s_i \cdot \lambda_i \mid s_i \in \{0, 1\}\}^n.$$

This set contains $\mathbb{Z}_2^n$ and thus a basis of $\mathbb{R}^n$. If we could guarantee additivity on pairs taken from $\mathcal{X}_0$ as well as scalar multiplicativity with respect to all scalars taken from Λ we could be sure to work with a correct linear function for our purposes.

Here a first problem occurs: For getting almost surely a linear function A on $\mathcal{X}_0$ from a table for A we need to know and test values of A on a much larger domain $\mathcal{X}_1$. So a larger test domain is needed in order to get a much smaller safe domain, compare [41]. The idea behind constructing $\mathcal{X}_1$ is as follows: We want $\mathcal{X}_1$ to be almost closed under

addition of elements from $\mathcal{X}_0$. With this we mean that for every fixed $x \in \mathcal{X}_0$, picking a random $y \in \mathcal{X}_1$ and adding x to it results with high probability again in an element in $\mathcal{X}_1$. Similarly, $\mathcal{X}_1$ should be almost closed under scalar multiplication with a factor $\lambda \in \Lambda$. These properties of $\mathcal{X}_1$ will be important in proving linearity of f_A on $\mathcal{X}_0$ if A satisfies the tests on $\mathcal{X}_1$ to be designed. Note that when we speak about linearity of f_A *on* $\mathcal{X}_0$ we mean that for all $x, y \in \mathcal{X}_0$ it holds $f_A(x) + f_A(y) = f_A(x+y)$, even though the sum $x+y$ in most cases does not belong to $\mathcal{X}_0$; similarly for arguments λx.

We remark that the above requirements are more difficult to be satisfied than in the corresponding construction of a long transparent proof in the Turing model. There, all domains are subsets of some $\mathbb{Z}_2^N$ and thus arguments are performed on a highly structured set with a lot of invariance properties of the uniform distribution. Secondly, there are no scalars other than 0 and 1, so additivity implies linearity. In the BSS setting some difficulties arise because some of the elements in Λ can be algebraically independent.

The above motivates the following definition. Let $M := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2\}\}$, $M^+ := \{\prod_{i=1}^K \lambda_i^{t_i} | t_i \in \{0, \ldots, n^2+1\}\}$ and

$$\mathcal{X}_1 := \left\{ \frac{1}{\alpha} \sum_{\beta \in M^+} s_\beta \cdot \beta \mid s_\beta \in \{0, \ldots, n^3\}, \alpha \in M \right\}^n .$$

We now prove that $\mathcal{X}_1$ does indeed have the desired properties. To keep things simple we will think of elements in $\mathcal{X}_0$, $\mathcal{X}_1$ (and later also in $\mathfrak{D}$) as formal sums of products defining M^+. This means for example that we distinguish elements in $\mathcal{X}_1$ which have the same numerical value because some λ_i's in Λ could be the same, but arise from formally different sums. Such elements are counted twice below when talking about the uniform distribution on the respective domains. Doing it this way simplifies some counting arguments because we don't have to take algebraic dependencies between the λ_i's into account.

Lemma 3.2. *Let $\epsilon > 0$ and let $n \in \mathbb{N}$ be large enough, then the following holds:*

a) For every fixed $x \in \mathcal{X}_0$ it is $\Pr_{y \in \mathcal{X}_1}\{y + x \in \mathcal{X}_1\} \geq 1 - \epsilon$.

Here, the probability distribution is the uniform one on $\mathcal{X}_1$, taking into account the above mentioned way how to count elements in $\mathcal{X}_1$.

b) Similarly, for fixed $\lambda_s \in \Lambda$ it is $\Pr_{y \in \mathcal{X}_1}\{\lambda_s \cdot y \in \mathcal{X}_1\} \geq 1 - \epsilon$.

c) For fixed $\lambda \in \Lambda$ it is $\Pr_{\alpha \in M}\{\alpha/\lambda \in M\} \geq 1 - \epsilon$.

Proof. For part a) let us focus on a single coordinate j. Then x_j is a 0-1 sum of the λ_i's. We have y_j of the form $\frac{1}{\alpha}\sum_{\beta \in M^+} s_\beta \cdot \beta$ with $\alpha \in M$ and $s_\beta \leq n^3$ for $\beta \in M^+$. If the sum for x_j contains a term $1 \cdot \lambda_i$ and the corresponding coefficient of monomial λ_i in y_j is $< n^3$, then $y_j + x_j$ also has the required form. Since $K = O(n)$ let $K \leq cn$ for a suitable constant $c > 0$. Thus for each of the at most K many addends in x_j there are n^3 out of $n^3 + 1$ choices for the coefficient of the corresponding monomial in y_j that imply $x_j + y_j$ to be of the required form with respect to this monomial. Since this argument applies for all n components one obtains

$$\Pr_{y\in\mathcal{X}_1}\{y+x\in\mathcal{X}_1\}=\left(\frac{n^3}{n^3+1}\right)^{K\cdot n}=\left(1-\frac{1}{n^3+1}\right)^{O(n^2)}\underbrace{\geq}_{Bernoulli}1-\frac{O(n^2)}{n^3+1}\geq 1-\frac{c}{n}\geq 1-\epsilon.$$

For part b) consider an arbitrary fixed $\lambda_s\in\Lambda$ together with a random $y\in\mathcal{X}_1$. Consider again a fixed component j of y. The α in the representation of this y_j has the form $\prod_{i=1}^{K}\lambda_i^{t_i}$ with $t_i\in\{0,\ldots,n^2\}$. If the particular exponent t_s of λ_s in this α satisfies $t_s>0$, then $\lambda_s\cdot y$ will belong to $\mathcal{X}_1$ (and for some cases with $t_s=0$ as well). The probability that $t_s>0$ and thus $\lambda_s\cdot y\in\mathcal{X}_1$ is therefore bounded from below by

$$\Pr_{y\in\mathcal{X}_1}\{\lambda_s\cdot y\in\mathcal{X}_1\}=\left(\frac{n^2}{n^2+1}\right)^{n}\geq 1-\frac{c}{n}\geq 1-\epsilon.$$

Part c) is trivial. □

In order to verify (almost) linearity of A on $\mathcal{X}_0$ with respect to scalars from Λ a test is designed that works on arguments of the forms $x+y$, where $x,y\in\mathcal{X}_1$ and $\alpha\cdot x$ with $\alpha\in M, x\in\mathcal{X}_1$. The function value table expected from a proof therefore must contain values in all arguments from the set $\mathfrak{D}:=\{x+y|x,y\in\mathcal{X}_1\}\cup\{\alpha\cdot x|\alpha\in M,x\in\mathcal{X}_1\}$. In the next subsection a test is designed on $\mathfrak{D}$ that verifies with high probability linearity of A on $\mathcal{X}_0$.

3.2 The linearity test and self-correction

As in the previous section we will only describe how things work for the function $A:\mathfrak{D}\to\mathbb{R}$. In the ideal case this function A is linear and thus uniquely encodes the coefficient vector $a\in\mathbb{R}^n$ of the related linear function.

In order to make the formulas look a bit simpler we define the abbreviation $A_\alpha(x):=A(\alpha\cdot x)/\alpha$. We repeat the following test a constant number of times:

Linearity test:

a) Uniformly and independently choose random x,y from $\mathcal{X}_1$ and random α,β from M;

b) check if $A(x+y)=A_\alpha(x)+A_\beta(y)$.

If all checks were correct the test accepts. Otherwise the test rejects.

Each round will inspect at most three different proof components, namely $A(x+y), A(\alpha\cdot x)$ and $A(\beta\cdot y)$. Thus in finitely many rounds $O(1)$ components will be inspected.

Clearly the linearity test accepts any linear function A with probability 1. For any $\delta > 0$ and $\epsilon > 0$ we can choose the number of repetitions of the linearity test so large that if

$$\Pr_{x,y \in \mathcal{X}_1, \alpha, \beta \in M} \{A(x+y) = A_\alpha(x) + A_\beta(y)\} > 1 - \delta \tag{3.3}$$

does not hold, then the test rejects with probability $1 - \epsilon$. The following cases have to be analyzed. If the linearity test rejects the verifier rejects the proof and nothing more is required. So suppose the linearity test does not give an error. If (3.3) is not satisfied, i.e., in particular the function value table does not come from a linear function, the verifier would err. Luckily it is easy to show that the probability for this to happen is small. And according to the definition of the $\mathrm{PCP}_{\mathbb{R}}$ classes we are allowed to accept incorrect proofs with small probability. It remains to deal with the only more difficult situation: The linearity test accepts and (3.3) holds. This of course does not mean that all values in the table necessarily are the correct ones. If the verifier asks for a particular such value we must therefore guarantee that at least with high probability we can extract the correct one from the table. One can get around this problem by defining a so-called self-correction f_A on $\mathcal{X}_1$ which can be shown to be linear on $\mathcal{X}_0$. This self-correction looks as follows: For $x \in \mathcal{X}_1$ define

$$f_A(x) = \mathrm{Majority}_{y \in \mathcal{X}_1, \alpha \in M} \{A_\alpha(x+y) - A_\alpha(y)\}.$$

Hence $f_A(x)$ is the value that occurs most often in the multiset $\{A_\alpha(x+y) - A_\alpha(y) | y \in \mathcal{X}_1, \alpha \in M\}$. It could be the case that $A_\alpha(x+y)$ is not defined. If this happens we just do not count this 'value'.

Lemma 3.3. *Under the above assumptions the function f_A is linear on $\mathcal{X}_0$ with scalars from Λ, i.e., for all $v, w \in \mathcal{X}_0$ we have $f_A(v+w) = f_A(v) + f_A(w)$ and for all $x \in \mathcal{X}_0, \lambda \in \Lambda$ we have $f_A(\lambda \cdot x) = \lambda \cdot f_A(x)$.*

Proof. For arbitrary fixed $v \in \mathcal{X}_0$ and random $x \in \mathcal{X}_1$ by Lemma 3.2 it is $x + v \in \mathcal{X}_1$ with probability $\geq 1 - \epsilon$ assuming n is large enough. Since $x \mapsto x + v$ is injective and due to the use of the uniform distribution in (3.3) replacing x by $x + v$ in (3.3) gives

$$\Pr_{x,y \in \mathcal{X}_1, \alpha, \beta \in M} \{A(x+v+y) = A_\alpha(x+v) + A_\beta(y)\} > 1 - \delta - \epsilon.$$

Doing the same with y instead of x yields

$$\Pr_{x,y \in \mathcal{X}_1, \alpha, \beta \in M} \{A(x+v+y) = A_\alpha(x) + A_\beta(v+y)\} > 1 - \delta - \epsilon$$

and combining these two inequalities results in

$$\Pr_{x,y \in \mathcal{X}_1, \alpha, \beta \in M} \{A_\alpha(x+v) - A_\alpha(x) = A_\beta(v+y) - A_\beta(y)\} > 1 - 2\delta - 2\epsilon.$$

From this it follows that

$$\Pr_{x \in \mathcal{X}_1, \alpha \in M} \{f_A(v) = A_\alpha(x+v) - A_\alpha(x)\} \geq 1 - 2\delta - 2\epsilon. \tag{3.4}$$

Similarly, for a fixed $w \in \mathcal{X}_0$ one obtains

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(w) = A_\alpha(x+w) - A_\alpha(x)\} \geq 1 - 2\delta - 2\epsilon$$

and using again the fact that shifting a random $x \in \mathcal{X}_1$ by a fixed $v \in \mathcal{X}_0$ does not change the distribution too much we obtain

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(w) = A_\alpha(x+v+w) - A_\alpha(x+v)\} \geq 1 - 2\delta - 3\epsilon. \tag{3.5}$$

Using the above argument a third time, now with $v+w$ instead of v (and thus 2ϵ instead of ϵ) we get

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(v+w) = A_\alpha(x+v+w) - A_\alpha(x)\} \geq 1 - 2\delta - 4\epsilon. \tag{3.6}$$

Combining (3.4), (3.5) and (3.6) it follows

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(v+w) = f_A(v) + f_A(w)\} \geq 1 - 6\delta - 9\epsilon.$$

This is independent of both x and α, so the probability is either 0 or 1. Hence, choosing δ and ϵ small enough it will be 1 and the first part of the linearity condition is proved.

Concerning scalar multiplicativity let $e_i \in \mathbb{R}^n$ be a unit vector and $\lambda \in \Lambda$. Since $\lambda \cdot e_i \in \mathcal{X}_0$ one can apply Lemma 3.2 together with (3.4) to get

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(\lambda\cdot e_i) = A_{\alpha/\lambda}(\lambda\cdot e_i + \lambda\cdot x) - A_{\alpha/\lambda}(\lambda\cdot x)\} \geq 1 - 2\delta - 4\epsilon.$$

Since $A_{\alpha/\lambda}(\lambda\cdot e_i + \lambda\cdot x) - A_{\alpha/\lambda}(\lambda\cdot x) = \lambda(A_\alpha(e_i+x) - A_\alpha(x))$ and by (3.4)

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(e_i) = A_\alpha(e_i+x) - A_\alpha(x)\} \geq 1 - 2\delta - 2\epsilon$$

it follows that

$$\Pr_{x\in\mathcal{X}_1,\alpha\in M}\{f_A(\lambda\cdot e_i) = \lambda f_A(e_i)\} \geq 1 - 4\delta - 6\epsilon.$$

This is again independent of x and α, so choosing δ and ϵ small enough yields $f_A(\lambda\cdot e_i) = \lambda f_A(e_i)$. Finally, given additivity on $\mathcal{X}_0$ and scalar multiplicativity for scalars $\lambda \in \Lambda$ on the standard basis the claim follows. □

3.3 Checking consistency

If the function value tables for both A and B have been tested with high probability to be close to unique linear functions f_A and f_B it remains to deal with consistency of these two functions. If $a \in \mathbb{R}^n, b \in \mathbb{R}^{n^2}$ are the corresponding coefficient vectors consistency means that $b_{ij} = a_i \cdot a_j$. In this subsection it is outlined how to test it.

For any $x \in \mathcal{X}_0$ and $\epsilon > 0$ it has been shown how to compute the correct value of $f_A(x)$ with probability $1-\epsilon$ by making only a constant number of queries. We can therefore from now on pretend to simply get the correct values of $f_A(x)$ and $f_B(z)$. The probabilities of obtaining an incorrect value at the places where these functions are used are added to the small probability with which we are allowed to accept incorrect proofs.

For $x \in \mathbb{R}^n$, let $x \otimes x$ denote the vector $y \in \mathbb{R}^{n(n+1)/2}$ for which $y_{i,j} = x_i \cdot x_j$, $1 \leq i \leq j \leq n$. Now a is consistent with b if and only if for all $x \in \mathbb{Z}_2^n$ it is the case that $f_A(x)^2 = f_B(x \otimes x)$.[1] This is the property that will be tested. Repeat the following consistency test a constant number of times:

Consistency test:

a) Uniformly choose random x from $\mathbb{Z}_2^n$;

b) check if $f_A(x)^2 = f_B(x \otimes x)$.

If in every round of the test the check is correct the verifier accepts, otherwise it rejects.

As with the linearity test the interesting case to deal with is when the verifier accepts the consistency test with high probability.

Lemma 3.4. *With the above notations if the consistency test accepts with probability $> \frac{3}{4}$, then consistency of a and b holds.*

Proof. The proof basically relies on the fact that if two vectors in some $\mathbb{R}^N$ are different multiplying both with a random $x \in \mathbb{Z}_2^N$ will give different results with probability at least $\frac{1}{2}$. This is applied to the two linear functions on $\mathbb{R}^{n^2}$ resulting from $a \otimes a$ and b. The same is true over $\mathbb{C}$. For details see [35]. □

3.4 Putting everything together

The linearity and consistency tests together ensure that any proof for which the self-corrections f_A and f_B are not linear on $\mathcal{X}_0$ or are not consistent are rejected with high probability. So the only thing left to do is to verify whether a is indeed a zero of the polynomial system. This is done by evaluating equation (3.2). If it evaluates to zero the verifier accepts, otherwise not.

Summarizing the results of this section we finally get the following theorem. Due to the fact that the verifier uses a proof of doubly exponential length in the theorem's statement we slightly deviate from the properties of a verifier as given in Definition 2.15. This is of no major concern as will be commented on after the theorem.

[1] The appropriate domain on which f_B can be shown to be linear in particular contains $x \otimes x$ for all $x \in \mathbb{Z}_2^n$.

Theorem 3.5. *For every problem $L \in \mathrm{NP}_{\mathbb{R}}$ there is a verifier working as follows: Given an instance w of size n the verifier expects a proof of length $f(n)$, where f is doubly exponential in n. The verifier generates uniformly a finite number of random strings. Using those strings it computes the addresses of constantly many proof-components it wants to read. This computation is done without reading the input w, i.e., the components to be seen only depend on the random strings generated. In its decision phase the verifier uses input w together with the finitely many components and accepts L according to the requirements of Definition 2.15. It has a decision time that is polynomially bounded in the input size n.*

The according statement holds for $\mathrm{NP}_{\mathbb{C}}$.

Let us comment on the theorem in view of Remark 3.1 above. Recall that the size of $\mathfrak{D}$ in our proof is doubly exponential in the input size n. Therefore, the random strings used in the proof above are exponential in length. In the verification procedure they are used to compute the proof-components which the verifier wants to see. In contrast to Definition 2.15 these components are computed independently of the concrete input w (but dependent on n). The reason to require this is that we want to forbid the verifier to potentially use exponential time in the query phase in order to decide the input. After having read the values of the finitely many components the verifier uses the input and the values of those components (and not any longer the random string) in order to make its decision after a running time being polynomial in the size of the input. The verifier constructed above thus is more restricted than general verifiers because it is limited with respect to how it computes the components to be seen.

Note however that the decisive point behind Theorem 3.5 is the structure of the verification proof. In the following chapters we shall see that for the full $\mathrm{PCP}_{\mathbb{R}}$ theorem transparent long proofs are invoked in a situation where inputs are of constant (combinatorial proof) or $O(\mathrm{polylogloglog}(n))$ (algebraic proof) size. In this situation of course also the length of each random string remains constant or logarithmic respectively. Then the structure of the verification procedure is more important than the parameter values; the latter automatically remain small enough. Therefore, when used in the framework of the full PCP theorem the verifier in Theorem 3.5 can again be chosen according to Definition 2.15. In these application of Theorem 3.5 the decision of whether the input QPS instance is satisfiable is not important (it will always be satisfiable). What is important is that it checks with a constant number of queries whether the given proof is 'close' to an encoding of a solution to the input equations.

The proof of Theorem 3.5 can be adapted word by word for the complex number BSS model. There is no argument involved that uses the presence of an ordering, except that in the definition of $P(x, r)$ we avoided to use instead the sum of the squared single polynomials of the system as could be done over the reals. This would save some small amount of randomness, introducing at the same time a more complicated polynomial of degree 4.

Chapter 4

Testing low-degree trigonometric polynomials

In this chapter we prove a theorem about testing low-degree trigonometric polynomials which will play a major role in the algebraic proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem in the next chapter. We decided to put this proof in a separate chapter for two reasons. The first reason is that the proof is long and contains many tedious details. The second reason is that we also find the testing of meaningful mathematical objects (in this case trigonometric polynomials) interesting in itself. In this chapter we will closely follow the presentation in [7].

Before we get into the details, let us first present an overview of the difficulties designing such a test in comparison with the classical situation. The crucial progress made in [2] with respect to a low-degree test is that given a function value table for $f : F^k \mapsto F$ for a finite field F the test is designed to perform certain checks along arbitrary straight lines through F^k. In order to make this idea working the structure of a finite field is essential with respect to both the resources needed and the probability arguments. There seem to arise major difficulties if one tries to generalize those ideas to the real numbers, i.e., when $F \subset \mathbb{R}$ is not any longer structured. The test performed in [36] uses paraxial lines and does not obey the structure required in order to use it as part of a verifier composition. It is unclear how to generalize it. One reason for this is the lacking structure of lines. Seeing F^k as subset of $\mathbb{R}^k$ will imply that a real line through F^k will leave the set. So one likely would have to enlarge the domain on which the initial function value table is given. But direct approaches seem to require much too large domains then. Note that a similar effect occurred in the previous chapter. However, this is not a problem due to the way this result will enter into the proofs of the $\mathrm{PCP}_{\mathbb{R}}$ theorem in Chapters 5 and 6.

A major goal we try to follow in our approach is to still make use of the properties of lines through an F^k, where F is a finite field. The solution we follow is to use other coding objects than algebraic polynomials for vectors in $\mathbb{R}^n$, namely multivariate trigonometric polynomials: The latter should, for a finite field F, map the set F^k to $\mathbb{R}$. The period of such trigonometric polynomials is taken as the field's cardinality q. This has the huge advantage

that in the domain it does not matter whether we consider arguments as real numbers or as finite field elements. As nice consequence, straight lines through F^k correspond to lines in $\mathbb{R}^n$ modulo that period. Though this gives back at least some of the advantages dealing with finite fields new difficulties arise that way. The major drawback is the following. All the above mentioned tests rely on restricting the given function table to one-dimensional subsets, thus working with univariate polynomials during the test. However, in contrast to algebraic polynomials the degree of a univariate restriction of such a trigonometric polynomial to an arbitrary line in F^k is not bounded by its original multivariate degree. Depending on the line chosen the degree of such a restriction can grow too much. This implies that not all lines are appropriate as restrictions to work with. As consequence, the design of a suitable set $H \subset F^k$ of directions of test-lines and the analysis of a corresponding test require considerable additional technical efforts. The latter are twofold. First, using the theory of expander graphs one has to establish that the set H is small, but still rich enough to cover in a reasonable sense all F^k. Secondly, it must be shown that a function table which does not give errors on H with high probability is close to a trigonometric polynomial on F^k.

As main result we obtain a verification procedure for trigonometric polynomials that inspects a constant number of relatively small blocks of proof components, thus giving a low-degree test which respects the structural requirements necessary for verifier composition. Independently of this aspect, we extend the still small list of interesting real number properties for which a probabilistic verification is possible. In particular, as far as we know trigonometric polynomials have not yet been used in the realm of real number complexity theory. Given the huge importance of Fourier analysis this might be interesting to be studied further.

The chapter is organized as follows. Section 4.1 describes the main task of testing whether a table of real values arises from such a polynomial is described precisely and a test to figure this out is given. The rest of the chapter then is devoted to prove that the test has the required properties. Towards this aim two major theorems have to be shown; this is done in Section 4.2.

4.1 Problem task and main result

In this section we will describe a probabilistic test performed by a real verifier to check whether a given multivariate function is close to a trigonometric polynomial of low degree. In the following sections we show the main result which specifies query complexity and success probability of the verifier.

Let us start with defining the main objects of this chapter, namely trigonometric polynomials. Let F be a finite field with $q := |F|$ being prime. As usual, we identify F with $\{0, \ldots, |F| - 1\}$. We consider particular real valued functions defined on some F^k.

Definition 4.1. *a) Let F be a finite field as above with cardinality q. For $d \in \mathbb{N}$ a*

univariate trigonometric polynomial of degree d from F to $\mathbb{R}$ is a function f of form

$$f(x) = a_0 + \sum_{m=1}^{d} a_m \cdot \cos(2\pi m \frac{x}{q}) + b_m \cdot \sin(2\pi m \frac{x}{q}),$$

where $a_0, \ldots, a_d$ and $b_1, \ldots, b_d$ are elements in $\mathbb{R}$.

b) For $k \in \mathbb{N}$ a multivariate trigonometric polynomial $f : F^k \mapsto \mathbb{R}$ of max-degree d is defined recursively via

$$\begin{aligned} f(x_1, \ldots, x_k) &= a_0(x_1, \ldots, x_{k-1}) + \sum_{m=1}^{d} a_m(x_1, \ldots, x_{k-1}) \cdot \cos(2\pi m \tfrac{x_k}{q}) + \\ &\quad + \sum_{m=1}^{d} b_m(x_1, \ldots, x_{k-1}) \cdot \sin(2\pi m \tfrac{x_k}{q}), \end{aligned}$$

where the a_i, b_j are trigonometric polynomials of max-degree d in $k-1$ variables.

Alternatively, one can write

$$f(x_1, \ldots, x_k) = \sum_{t} c_t \cdot exp(2\pi i \sum_{j=1}^{k} x_j t_j),$$

where the sum is taken over all $t := (t_1, \ldots, t_k) \in \mathbb{Z}^k$ with $|t_1| \leq d, \ldots, |t_k| \leq d$ and $c_t \in \mathbb{C}$ satisfy $c_t = \overline{c_{-t}}$ for all such t.

In all situations below the potential degrees we work with will be much less than the field's cardinality.

Since we shall mainly deal with trigonometric polynomials we drop most of the times the term 'trigonometric'. Whenever we refer to usual algebraic polynomials we state it explicitly.

The ultimate goal of this chapter is to design a verifier which performs a test whether a given table of function values is generated with high probability by a multivariate polynomial. More precisely, the following is the main result of this chapter.

Theorem 4.2 (Low-degree test)**.** *Let $d \in \mathbb{N}$, $h = 10^{15}$, $k \geq \frac{3}{2}(2h+1)$ and let F be a finite field with $q := |F|$ being a prime number larger than $10^4(2hkd+1)^3$. We fix these values for the entire chapter. There exists a probabilistic verification algorithm in the BSS-model of computation over the reals with the following properties:*

1. *The verifier gets as input a function value table of a multivariate function $f : F^k \to \mathbb{R}$ and a proof string π consisting of at most $|F|^{2k}$ segments (blocks). Each segment consists of $2hkd + k + 1$ real components.*

   The verifier first uniformly generates $O(k \cdot \log q)$ random bits; next, it uses the random bits to determine a point $x \in F^k$ together with one segment in the proof string it wants to read. Finally, using the values of $f(x)$ and those of the chosen

segment it performs a test (to be described below). According to the outcome of the test the verifier either accepts or rejects the input.

The running time of the verifier is polynomially bounded in the quantity $k \cdot \log q$, i.e., polylogarithmic in the input size $O(k \cdot q^{2k})$.

2. *For every function value table representing a trigonometric max-degree d polynomial there exists a proof string such that the verifier accepts with probability* 1.

3. *For any $0 < \epsilon < 10^{-19}$ and for every function value table whose distance to a closest max-degree $2hkd$ polynomial is at least 2ϵ the probability that the verifier rejects is at least ϵ, no matter what proof string is given. Here, for two functions $f, g : F^k \mapsto \mathbb{R}$ their distance is defined as $dist(f,g) := \frac{1}{|F^k|} \cdot |\{x \in F^k | f(x) \neq g(x)\}|$.*

The first and the second property in the theorem will follow directly from the description of the test. Proving the last property - as usual with such statements - is the main task. Repeating the verifier's computation constantly many times decreases the error probability below any given fixed positive constant.

Performing (one round of) the test the verifier reads $2hkd+k+1$ real numbers. Thus, it can only test for a local property of low-degree polynomials $f : F^k \mapsto \mathbb{R}$. A major amount of work will be to figure out what this local property should look like. The starting idea is common for low-degree tests, namely to consider univariate restrictions along certain lines of F^k. The segments of a proof string mentioned above precisely present the coefficients of such a univariate restriction. An advantage using a finite field as domain is that such lines only contain $|F|$ many points. So we do not have to deal with the problem of splitting the domains into a large test domain and a small safe domain as it is, for example, the case with the real linearity test from the previous chapter. On the other hand, it will turn out not to be a good idea to allow any arbitrary line in F^k for such a test as it is done in the classical approach [2]. A fair amount of work will be necessary to figure out a suitable subset $H \subset F^k$ of lines for which the test shall be performed.

4.1.1 Appropriate domain of test directions; structure of a verification proof

As mentioned above, the verifier expects each segment of the proof to specify a univariate polynomial of *appropriate* degree on a line. Since univariate restrictions of trigonometric polynomials along a line behave a bit differently than univariate restrictions of algebraic polynomials some care is necessary. Let $f : F^k \mapsto \mathbb{R}$ be a (trigonometric) max-degree d polynomial and let $\ell := \{x + tv | t \in F\}, x, v \in F^k$ be a line. For determining an upper bound of the degree of the univariate restriction of f on ℓ it turns out to be helpful to define a kind of absolute value for the elements of F. The definition is inspired by the fact that if we later on restrict a trigonometric polynomial to lines with small components in absolute value the resulting univariate polynomials have a relatively small degree.

Definition 4.3 (Absolute value)**.** *For $t \in F = \{0, \ldots, |F|-1\}$ put*

$$|t| = \begin{cases} t & \text{if } t < |F|/2 \\ |F| - t & \text{if } t > |F|/2 \end{cases} .$$

If a univariate polynomial $p : F \to \mathbb{R}$ has degree d, then for $a, b \in F$ the polynomial $t \mapsto p(a + bt)$ has degree at most $d \cdot |b|$ and thus $t \mapsto f(x + tv)$ has degree at most $d \cdot \sum_{i=1}^{k} |v_i|$, where $v = (v_1, \ldots, v_k)$. This is an easy consequence of Definition 4.1.

For the test performed by the verifier we want to specify a suitable set $H \subset F^k$ of lines along which univariate restrictions are considered. Suitable here refers to the maximal degree such a restriction could have given a max-degree d multivariate polynomial. This maximal degree in a certain sense should be small. The constant parameter h in Theorem 4.2 determines what we mean by small. Though $h = 10^{15}$ of course is a large constant the decisive point is its independence of d, k, q.

Definition 4.4. *Let F be a finite field, $k \in \mathbb{N}$ and h be as in Theorem 4.2. The set H is defined to be any subset of $F^k \setminus \{0\}$ satisfying the following two conditions:*

i) For every $0 \neq v := (v_1, \ldots, v_k) \in H$ it is $|v| := \max\{|v_1|, \ldots, |v_k|\} \leq h$ and

ii) if for a fixed $v \in F^k$ several points in the set $\{tv | t \in F\}$ satisfy condition i) only one of them is included in H.

Condition i) requires the direction of lines that we consider to have small components, whereas condition ii) just guarantees that each line (as point set) is included at most once. We abstain from specifying which v we include in such a case and just fix H as one such set.

If a k-variate polynomial of max-degree d is restricted to a line $\{x + tv | t \in F\}$ whose direction v belongs to H, then the resulting univariate polynomial has degree at most hkd. Note that for the values chosen hkd is much smaller than $\frac{|F|}{2}$. In later arguments the cardinality of H will be important, so let us say something about it already now. For h sufficiently smaller than $|F|$ there are $(2h+1)^k - 1$ elements $v \in F^k - \{0\}^k$ such that $|v| \leq h$. For every such v it is $|-v| \leq h$, therefore $\frac{1}{2}((2h+1)^k - 1)$ is an upper bound for $|H|$. We shall prove below in Lemma 4.22 that for increasing k the fraction $\frac{|H|}{\frac{1}{2}((2h+1)^k-1)}$ approaches 1.

Given a table of function values for an $f : F^k \mapsto \mathbb{R}$ the verifier now expects the following information from a potential proof that f is a trigonometric polynomial of max-degree d. For every line ℓ in F^k which has a direction $v \in H$ the proof should provide a segment of real numbers which represent a univariate polynomial as follows. The segment consists of a point $x \in F^k$ on the line as well as reals $a_0, \ldots, a_{hkd}, b_1, \ldots, b_{hkd}$. The verifier will interpret this information as the univariate polynomial with coefficients a_i, b_j that ideally, i.e., for a trigonometric polynomial, represents f's restriction to ℓ parametrized as $t \mapsto f(x + tv)$. Obviously, there are several different parametrizations depending on the point x, but we only code one. For the purposes below this is sufficient.

The total length of such a proof is easily calculated. The point $x \in F^k$ requires k reals, thus each segment in the proof needs $2hkd + k + 1$ reals. Each of the $|H|$ directions contributes $|F|^{k-1}$ different (parallel) lines[1], so the total number of segments is $|H| \cdot |F|^{k-1}$ and the length of the proof is upper bounded by $(2hkd + k + 1) \cdot \frac{1}{2}((2h+1)^k - 1) \cdot |F|^{k-1}$. For our choices of parameters $k, d, |F|$ this is smaller than $|F|^{2k}$.

We are now prepared to describe the test in detail:

Low-degree test: Let F and H be as above.

Input: A function $f : F^k \to \mathbb{R}$, given by a table of its values; a list of univariate trigonometric polynomials defined for each line in F^k with direction in H and specified by its coefficients as described above.

1. Pick uniformly at random a direction $v \in H$ and a random point $x \in F^k$;
2. compute deterministically the unique segment in the proof that specifies the univariate polynomial $p_{v,x}(t)$ which is defined on the line through x in direction v; compute as well the unique $\tau_x \in F$ such that $p_{v,x}(\tau_x)$ is the value in point x;[2]
3. if $f(x) = p_{v,x}(\tau_x)$ accept, otherwise reject.

Since F is discrete the objects picked in Step 1 can be addressed using $O(k \cdot \log |F|)$ random bits. Note that this first step is the same as saying we pick a random direction $v \in H$ together with a random line among those having direction v. There are $|F|$ many points on each such line, i.e., $|F|$ choices for x result in the same line.

A few words are necessary due to the potential ambiguity of representing a line. Suppose the proof specifies a univariate polynomial $p_{v,x'}(t)$ on a line $\{x' + tv | t \in F\}$ for some x', v. If x is another point on the line we could of course use as well another parametrization. Nevertheless, given x and v it is no problem to compute the unique x' as well as τ_x such that $x = x' + \tau_x \cdot v$. Thus, the value the test is looking for to equal $f(x)$ is $p_{v,x'}(\tau_x)$. In order to avoid a notational overhead and with a slight abuse below we often identify the polynomials $p_{v,x'}$ and $p_{v,x}$ if x, x' lie on the same line in direction v. Similarly, evaluating one of them in τ_x is understood to take the correct parameter in order to evaluate the polynomial as given in the proof in x (seen as univariate evaluation, of course). Similarly, we often denote the corresponding lines both by $\ell_{v,x}$ or $\ell_{v,x'}$. There should be no danger of confusion.

[1] here we need that no element in H is a multiple of another one and that each line contains $|F|$ points

[2] in order to evaluate such a polynomial the verifier could use $\cos \frac{2\pi}{q}$ and $\sin \frac{2\pi}{q}$ as machine constants. For varying input sizes we address this point again in the final section.

4.2 Proof of Main Theorem

The proof of Theorem 4.2 relies on two major theorems together with several propositions and lemmas. Let us outline the main ideas behind those theorems. The first one is Theorem 4.5. It states that the rejection probability of the low-degree test is about proportional to the distance δ of the given function f to a closest max-degree $2hkd$ polynomial. However, this will hold only in case that this distance δ is not too big. For too large distances the lower bound which the theorem gives for the rejection probability gets very small (and even can become negative). So intuitively the theorem states that if the rejection probability is small, then f is either very close or very far away from a max-degree $2hkd$ polynomial.

Theorem 4.5. *Let δ denote the distance of a function $f : F^k \to \mathbb{R}$ to a closest max-degree $2hkd$ polynomial. Then for any proof the probability that the low-degree test rejects is at least*

$$\left(\frac{2h}{2h+1}(1-\delta) - \frac{4h^2k^2d}{|F|-1}\right)\delta.$$

Thus we have to deal with the case that though the rejection probability might be small given the above lower bound f is far away from such a polynomial. Theorem 4.6 basically shows that this case cannot occur using the following idea: if for a function f and a proof string π the probability of rejection is small, then f and π can be changed in a number of small steps such that these changes do not increase the rejection probability too much and in the end a max-degree $2hkd$ polynomial f_s is obtained. Since by Theorem 4.5 such a transformation process would not be possible if f would be far away from any max-degree $2hkd$ polynomial (the process would have to cross functions for which the test rejects with higher probability), it follows that a reasonably small rejection probability only occurs for functions f that were already close to a max-degree $2hkd$ polynomial.

Theorem 4.6. *Let $0 < \epsilon \leq 10^{-19}$ and let a function f_0 together with a proof π_0 be given. If the low-degree test rejects with probability at most ϵ, then there is a sequence $(f_0, \pi_0), (f_1, \pi_1), \ldots, (f_s, \pi_s)$ such that*

1. *for every $i \leq s$ the probability that the test rejects input (f_i, π_i) is at most 2ϵ,*
2. *for every $i < s$ the functions f_i and f_{i+1} differ in at most $|F|$ arguments and*
3. *the function f_s is a max-degree $2hkd$ polynomial.*

Note that it is only the existence of such a sequence that we need for the proof of Theorem 4.2, it does not describe anything that the test does. So it does not have to be (efficiently) computable.

Assuming validity of both theorems the Main Theorem can be proven as follows:

Proof. (of Main Theorem 4.2) Statements 1. and 2. of the theorem being obvious from the foregoing explanations let us assume we are given a function value table for $f : F^k \mapsto \mathbb{R}$

together with a verification proof π such that the low-degree test rejects with a probability at most $\epsilon \leq 10^{-19}$. We will show that the distance δ from f to the set of max-degree $2hkd$ polynomials is at most 2ϵ. In order to avoid tedious calculations the arguments are given on a slightly more abstract level based on continuity, however it is no problem to fill in the precise values for δ in all cases.

Since h is large and $|F| \gg 4h^2k^2d$ Theorem 4.5 reads $(c_1(1-\delta)-c_2)\cdot\delta \leq \epsilon$, where constant c_1 is close to 1 and c_2 is close to 0. This implies that either δ is at most slightly larger than ϵ or δ is close to 1. We want to exclude the second case. Assume for a contradiction that δ is close to 1. By Theorem 4.6 there exists a function f' and a verification proof π' such that the probability that the test rejects the pair (f', π') is at most 2ϵ and the distance δ' from f' to the closest max-degree $2hkd$ polynomial is close to $\frac{1}{2}$. This is true because each new element (f_i, π_i) in the sequence reduces the number of errors by at most $|F|$. Now choose the number of reducing steps such that the number of errors is reduced from $\delta|F|^k$ to approximately $\frac{1}{2}|F|^k$. This must happen because finally a polynomial is obtained. Again using Theorem 4.5 it follows that the test must reject (f', π') with a probability which is at least about $\frac{1}{4}$. This contradicts the fact that (f', π') is rejected with probability at most 2ϵ and we have reached the desired contradiction. □

4.2.1 Proof of Theorem 4.5

We shall now turn to the proofs of the two major theorems. Especially that for Theorem 4.6 needs considerable additional technical efforts. In order to streamline the presentation some of the proofs will be postponed to the final section.

Let us first prove Theorem 4.5. Suppose we are given a function $f : F^k \to \mathbb{R}$ with (exact) distance $\delta > 0$ to a closest max-degree $2hkd$ polynomial $\tilde{f}$. Let π denote an arbitrary verification proof specifying univariate polynomials $p_{v,x}$ as described above. Define the set $U \subset F^k$ as those points where f and $\tilde{f}$ disagree. Thus $|U| = \delta|F|^k$. The idea behind our analysis is to guarantee the existence of relatively (with respect to δ) many pairs $(x, y) \in U \times \bar{U}$ that are located on a line with direction $v \in H$. More precisely, we shall consider the set C of triples (x, v, y) with $x \in U, v \in H$, and $y \in \bar{U} = F^k \setminus U$ on the line $\ell_{v,x}$. The goal is to show that C contains many triples (x, v, y) for which $p_{v,x}(\tau_x) \neq f(x)$ or $p_{v,x}(\tau_y) \neq f(y)$. [3]

Due to $x \in U$ and $y \in \bar{U}$ for any $(x, v, y) \in C$ one of the following two alternatives holds:

1. The polynomial $p_{v,x}$ is different from $\tilde{f}$ on $\ell_{v,x}$ but agrees with $\tilde{f}$ in y.

2. The polynomial $p_{v,x}$ disagrees with f in x or in y.

[3] To avoid misunderstandings we point out once more that here we mean the value of the univariate polynomial $\tau \mapsto p_{v,x}(\tau)$ in the respective arguments τ_x and τ_y that on the corresponding line through x and y in direction v result in points x and y, respectively.

Claim: Alternative 1. is satisfied by at most $\delta|F|^k \cdot |H| \cdot 4h^2k^2d$ triples from C.

Proof of Claim: Since $\tilde{f}$ is a max-degree $2hkd$ polynomial, the restriction of $\tilde{f}$ to any line yields a univariate degree $2h^2k^2d$ polynomial. By usual interpolation for trigonometric polynomials different univariate polynomials of degree at most $2h^2k^2d$ agree in at most $4h^2k^2d$ points. Applying this argument to each of the $|H|$ lines and taking into account that there are $|U| = \delta|F|^k$ choices for x there can be at most

$$\delta|F|^k \cdot |H| \cdot 4h^2k^2d$$

triples in C satisfying the first alternative.

Next we aim for a lower bound on the total number of triples in C. This easily implies a lower bound on the number of triples satisfying the second alternative; from that a lower bound for the probability that the test rejects can be obtained.

A lower bound on $|C|$ is given in the following proposition. It basically says that the set H of test directions is sufficiently large. Since its proof requires considerable technical efforts relying on the theory of expander graphs we just give the statement here and postpone the proof to the final section.

Proposition 4.7. *Let $U \subseteq F^k$ as above (or any other arbitrary set with cardinality $\delta|F|^k$) with $0 \leq \delta \leq 1$. Then there are at least*

$$\frac{2h}{2h+1}\delta(1-\delta)|H|(|F|-1)|F|^k$$

pairs $(x, y) \in U \times (F^k - U)$ such that the line through x and y has direction in H.

This proposition together with the above claim implies that the number of triples in C satisfying the second alternative is at least

$$\frac{2h}{2h+1}\delta(1-\delta)|H|(|F|-1)|F|^k - \delta|F|^k \cdot |H| \cdot 4h^2k^2d.$$

In order to finish the proof of Theorem 4.5 an alternative view of the low-degree test helps. The test can as well be seen as first choosing randomly two points x, y such that they determine a direction $v \in H$. Since there are $|H|$ directions and $|F|$ points on each line there are $|H|(|F|-1)|F|^k$ such triples (x, v, y) in total. Then, with probability $\frac{1}{2}$ the test decides whether to check if $p_{v,x}(\tau_x) = f(x)$ or if $p_{v,x}(\tau_y) = f(y)$. Since triples in C that satisfy alternative 2 result in an error for the low-degree test if the appropriate point for evaluation is chosen, its probability for rejection is at least

$$\left(\frac{2h}{2h+1}(1-\delta) - \frac{4h^2k^2d}{|F|-1}\right)\delta.$$

Half of this value is contributed by triples $(x, v, y) \in C$ for which $p_{v,x}(\tau_x) \neq f(x)$ or $p_{v,x}(\tau_y) \neq f(y)$. The other half arises from triples (y, v, x) for which $(x, v, y) \in C$ and $p_{v,x}(\tau_x) \neq f(x)$ or $p_{v,x}(\tau_y) \neq f(y)$. □

4.2.2 Proof of Theorem 4.6

In order to prove Theorem 4.6 reducing the inconsistency of a proof string plays an important role. This inconsistency is defined as follows.

Definition 4.8. *The* inconsistency *of a proof string π is the fraction of triples $(x, v, v') \in F^k \times H \times H$ for which $p_{v,x}(\tau_x) \neq p_{v',x}(\tau_x)$.*

The attentive reader might wonder that in the definition *ordered* triples are counted and that $v = v'$ is not excluded, so the inconsistency can never equal 1. These are only technical issues to make some calculations below easier.

In proving the existence of a sequence $\{(f_i, \pi_i)\}_i$ as in the theorem's statement the main idea is to find for a current pair (f, π) a segment in π that can be changed in a way which decreases the inconsistency. After that, also f is changed to a function f' which fits the best to the new proof string π'. The best fit is defined using majority decisions.

Definition 4.9. *Let π be a proof string for verifying a low-degree polynomial as explained above. As usual, denote by $\{p_{v,x} | v \in H, x \in F^k\}$ the univariate restrictions given with π. A* majority function *for π is a function $f : F^k \to \mathbb{R}$ such that for all $x \in F^k$ the value $f(x)$ maximizes $|\{v \in H | p_{v,x}(\tau_x) = f(x)\}|$.*

Ties can be broken arbitrarily in the definition of a majority function, so it does not necessarily have to be unique.

When proving Theorem 4.6 the sequence $(f_0, \pi_0), \ldots, (f_s, \pi_s)$ is defined by first changing π_i to π_{i+1} on a single proof segment and then taking f_{i+1} as a majority function for π_{i+1}. During this process there should occur no pair (f_i, π_i) for which the rejection probability is more than twice as large as the rejection probability of the initial (f_0, π_0). The following easy lemma will be useful in this context. It relates the rejection probability of the low-degree test for a pair (f, π) with inconsistency of π.

Lemma 4.10. *If f is a majority function for π, then the inconsistency of π is at least as large as the rejection probability of the low-degree test for (f, π) and at most twice as large.*

Proof. For a fixed $x \in F^k$ let m_x be the multiplicity with which the majority value for $f(x)$ occurs with respect to the values $p_{v,x}(\tau_x)$ specified by π. Thus, for m_x pairs (x, v) the equation $p_{v,x}(\tau_x) = f(x)$ holds. The rejection probability is $1 - \frac{\sum_x m_x}{|F^k| \cdot |H|}$. Now for any $v \in H$ there cannot be more than m_x many $v' \in H$ such that (x, v, v') satisfies $p_{v,x}(\tau_x) = p_{v',x}(\tau_x)$. Applying this to the $|H|$ many directions v there are at most $|H| \cdot \sum_x m_x$ triples that do not contribute to π's inconsistency. Rearranging shows that the inconsistency is at least as large as the rejection probability.

Vice versa, for fixed x there must be an $H' \subset H$ with $|H'| = m_x$ such that for all $v, v' \in H'$ the m_x^2 many equations $p_{v,x}(\tau_x) = p_{v',x}(\tau_x)$ are satisfied. Thus the inconsistency is upper bounded by $1 - \frac{\sum_x m_x^2}{|F^k| \cdot |H|^2}$. The latter is easily shown to be at most twice the rejection probability, i.e., $2 \cdot \left(1 - \frac{\sum_x m_x}{|F^k| \cdot |H|}\right)$. □

Without loss of generality we may assume that for the inputted pair (f_0, π_0) function f_0 is already a majority function for π_0. If this would not be the case we could just define the first pair in the sequence of (f_i, π_i) by changing stepwise one value of f_0 while leaving proof string π_0 unchanged until we have reached a majority function of π_0. Clearly, during this process the rejection probability will not increase.

Now in each further step the inconsistency of a current proof string π_i is strictly reduced for π_{i+1}. This is achieved by changing only one of π_i's segments. Furthermore, the new function f_{i+1} is obtained from f_i in such a way that it becomes a majority function for π_{i+1}. Since inconsistency is reduced step by step Lemma 4.10 implies that for every $i \leq s$ the rejection probability of (f_i, π_i) can be at most twice as large as the rejection probability of (f_0, π_0). Of course, we have to guarantee that the way the f_i's are changed finally turn them into a trigonometric polynomial.

The following proposition is a key stone to make the above idea precise.

Proposition 4.11. *Let π be a proof string obeying the general structure required for the low-degree test and having inconsistency at most 2ϵ, where $\epsilon \leq 10^{-19}$. Let f be a majority function of π which is not already a max-degree $2hkd$ polynomial.*

Then there exists a proof string π' such that π' differs from π in only one segment and its inconsistency is strictly less than that of π.

The proof needs several additional technical results which are proved in the final section. Let us first collect them and then prove the proposition. The following definition specifies certain point sets important in the further analysis.

Definition 4.12. *Let a pair (f, π) as above be given. Let $\alpha := 10^{-2}$ for the rest of the chapter.*

a) Define $S \subseteq F^k$ to consist of those points x for which the fraction of directions $v \in H$ satisfying $p_{v,x}(\tau_x) = f(x)$ is less than $1 - \alpha$.

b) For $v \in H$ define $S(v) \subseteq F^k$ as $S(v) := \{x \in F^k | x \notin S \text{ and } p_{v,x}(\tau_x) \neq f(x)\}$.

The set S contains those points for which there are relatively many, namely at least $\alpha|H|$, inconsistencies between different line polynomials through x and the value $f(x)$. The set $S(v)$ on the other hand contains the points for which most of the line polynomials agree with f on x, but the particular $p_{v,x}$ does not. As consequence, the latter disagrees with most of the others with respect to point x.

The main purpose using the following proposition is to pick out a line along which the given proof can be changed in such a way that its inconsistency reduces. For obtaining this line ℓ_{v^*,x^*} the objects x^* and v^* are determined by the following crucial proposition proved in the the final section of this chapter.

Proposition 4.13. *Let π be a proof string as in Proposition 4.11. There exist $x^* \in F^k$, $v^* \in H$ and a set $H' \subseteq H$ such that*

1. *$x^* \in S(v^*)$;*
2. *at most $\frac{1}{40}\alpha \cdot |F|$ points on ℓ_{v^*,x^*} belong to S;*
3. *$|H'| \geq (1-4\alpha)|H|$ and*
4. *for all $v \in H'$*
   - *i) the fraction of pairs $(t,s) \in F^2$ for which $p_{v^*,x^*+sv}(\tau_{x^*+tv^*+sv}) \neq p_{v,x^*+tv^*}(\tau_{x^*+tv^*+sv})$ is at most $\frac{1}{4}$ and*
   - *ii) the fraction of $s \in F$ for which $p_{v^*,x^*+sv}(\tau_{x^*+sv}) \neq p_{v,x^*}(\tau_{x^*+sv})$ is at most $\frac{1}{2}$.*

The second technical result that we need is a direct adaption of a similar lemma by Arora and Safra [3] to trigonometric polynomials. It says that if the entries of an $|F| \times |F|$ matrix both row-wise and column-wise arise to a large extent from univariate polynomials, then the majority of values of the entire matrix arise from a bivariate polynomial. For sake of completeness we will as well include a proof in the final section.

Lemma 4.14. *(see [2], adapted for trigonometric polynomials) Let $\tilde{d} \in \mathbb{N}, |F| \geq 10^4(2\tilde{d}+1)^3$. Suppose there are two sets of univariate trigonometric degree $\tilde{d}$ polynomials $\{r_s\}_{s\in F}$ and $\{c_t\}_{t\in F}$ such that the fraction of pairs $(s,t) \in F^2$ for which there is a disagreement, i.e., $r_s(t) \neq c_t(s)$, is at most $\frac{1}{4}$. Then there exists a bivariate trigonometric max-degree $\tilde{d}$ polynomial $Q(s,t)$ such that for at least a $\frac{2}{3}$-fraction of rows s it holds that $r_s(t) \equiv Q(s,t)$; similarly for at least a $\frac{2}{3}$-fraction of columns t it holds that $c_t(s) \equiv Q(s,t)$.*

Having all technical auxiliary material at hand we can now prove Proposition 4.11 and Theorem 4.6.

Proof. (of Proposition 4.11) Let x^*, v^* and H' be fixed according to Proposition 4.13. The segment we are going to change in π is the segment claiming a univariate polynomial on line ℓ_{v^*,x^*}. We need to show that this can be done in a way that decreases inconsistency.

We want to apply Lemma 4.14 to every (v^*, v')- plane through x^*, where $v' \in H'$. For such a v' define

$$\begin{aligned} r_s^{(v')}(t) &:= p_{v^*,x^*+sv'}(\tau_{x^*+tv^*+sv'}) \\ c_t^{(v')}(s) &:= p_{v',x^*+tv^*}(\tau_{x^*+tv^*+sv'}). \end{aligned}$$

Proposition 4.13, item 4,i) implies that the assumptions of Lemma 4.14 are satisfied for each v'. Let $Q^{(v')}$ denote the corresponding bivariate polynomial of max-degree $\tilde{d} := hkd$.

Note that every (v^*, v')-plane through x^* contains the line ℓ_{v^*,x^*} and that for every $v' \in H'$ it is $r_0^{(v')}(t) = p_{v^*,x^*}(\tau_{x^*+tv^*})$ independently of v'. Thus we abbreviate $r_0 := r_0^{(v')}$. The idea now is to show that there exists a degree hkd polynomial $R : F \to \mathbb{R}$ such that

R is different from r_0 and for most $v' \in H'$ the function $t \mapsto c_t^{(v')}(0)$ is close to R. From the precise version of this statement it will then follow that changing in π the segment containing r_0 to R will decrease the inconsistency.

For $v' \in H'$ let $R^{(v')}(t) := Q^{(v')}(0,t)$. We want to show that for many $v', v'' \in H', R^{(v')} \equiv R^{(v'')}$. The majority of these polynomials then defines R.

Claim 1: Let $v', v'' \in H'$. If $R^{(v')} \not\equiv R^{(v'')}$, then the distance between $t \mapsto c_t^{(v')}(0)$ and $t \mapsto c_t^{(v'')}(0)$ is at least

$$\frac{1}{3} - \frac{2hkd}{|F|}.$$

Proof of Claim 1: By Lemma 4.14 $R^{(v')}$ is the unique polynomial to which the function $t \mapsto c_t^{(v')}(0)$ is close with a distance of at most $\frac{1}{3}$. If $R^{(v')} \not\equiv R^{(v'')}$, then as polynomials of degree hkd they differ in at least $|F| - 2hkd$ points, thus $t \mapsto c_t^{(v')}(0)$ and $t \mapsto c_t^{(v'')}(0)$ have at least the claimed distance.

Next consider the number of inconsistencies on ℓ_{v^*,x^*}, i.e., the number of triples $(y, v, w) \in \ell_{v^*,x^*} \times H^2$ for which $p_{v,y}(\tau_y) \neq p_{w,y}(\tau_y)$. Proposition 4.13 intuitively implies that the number of inconsistencies cannot be too large. On the other hand, Claim 1 above implies that any two v', v'' for which $R^{(v')} \not\equiv R^{(v'')}$ will lead to many inconsistencies on ℓ_{v^*,x^*}. Hence, for most $v', v'' \in H'$ it will be the case that $R^{(v')} \equiv R^{(v'')}$. More precisely:

Claim 2: The number of pairs $(v', v'') \in (H')^2$ for which $R^{(v')} \equiv R^{(v'')}$ is at least

$$\left((1-4\alpha)^2 - \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}} \right) \cdot |H|^2. \tag{4.1}$$

Proof of Claim 2: The lower bound is obtained by comparing the number of inconsistencies caused by triples $(y, v, w) \in \ell_{v^*,x^*} \times H^2$ on the one hand side and those caused by triples $(y, v', v'') \in \ell_{v^*,x^*} \times (H')^2$ where $R^{(v')} \not\equiv R^{(v'')}$ on the other. We restrict y to belong to $\ell_{v^*,x^*} \cap F^k \setminus S$ (recall Definition 4.12) and give an upper bound on the first quantity and a lower bound on the second that allows to conclude the claim.

For any $y \notin S$ there are at least $(1-\alpha)|H|$ directions $w \in H$ such that the values $p_{w,y}(\tau_y)$ coincide with $f(y)$ and thus with each other; so for such a fixed y at least $(1-\alpha)^2|H|^2$ triples will not result in an inconsistency. Vice versa, at most $(1-(1-\alpha)^2)|H|^2 \leq 2\alpha|H|^2$ inconsistencies can occur. Since there are at most $|F|$ choices for $y \in \ell_{v^*,x^*}$ we have the following upper bound:

$$|\{(y, v, w) \in \ell_{v^*,x^*} \times H^2 | y \notin S \text{ and } p_{v,y}(\tau_y) \neq p_{w,y}(\tau_y)\}| \leq 2\alpha|H|^2|F|. \tag{4.2}$$

Next consider inconsistencies (y, v', v'') caused by $(v', v'') \in H'$ such that $R^{(v')} \not\equiv R^{(v'')}$. According to Claim 1 each such pair (v', v'') implies the existence of at least $\frac{1}{3}|F| - 2hkd$ points $y \in \ell_{v^*,x^*}$ such that (y, v', v'') is an inconsistency for π. Requiring in addition $y \notin S$ according to Proposition 4.13 will still give at least $\frac{1}{3}|F| - 2hkd - \frac{1}{40}\alpha|F|$ many such y, i.e., for each $(v', v'') \in H', R^{(v')} \not\equiv R^{(v'')}$ it holds

$$|\{y \in \ell_{v',v''} | y \notin S \text{ and } p_{v',y}(\tau_y) \neq p_{v'',y}(\tau_y)\}| \geq \frac{1}{3}|F| - 2hkd - \frac{1}{40}\alpha|F|. \tag{4.3}$$

Combining (4.2) and (4.3) it follows that the number of pairs $(v', v'') \in (H')^2$ for which $R^{(v')} \not\equiv R^{(v'')}$ is upper bounded by

$$\frac{2\alpha|H|^2|F|}{\frac{1}{3}|F| - 2hkd - \frac{1}{40}\alpha|F|} = \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}} \cdot |H|^2.$$

Since $|H'| \geq (1-4\alpha)|H|$ this in turn means that the number of pairs $(v', v'') \in (H')^2$ for which $R^{(v')} \equiv R^{(v'')}$ must be at least

$$\left((1-4\alpha)^2 - \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}}\right) \cdot |H|^2. \tag{4.4}$$

This yields Claim 2.

Next, define the univariate polynomial R as the majority polynomial among all $R^{(v')}$, i.e., the polynomial which maximizes $|\{v' \in H' | R^{(v')} \equiv R\}|$.

Claim 3: The number of choices $v' \in H'$ such that $R \equiv R^{(v')}$ is at least $\beta|H|$, where

$$\beta \geq (1-4\alpha)^2 - \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}} > 0.84.$$

Proof of Claim 3: Let β be the fraction in H (not in H'!) of directions v' which belong to H' and satisfy $R^{(v')} \equiv R$, i.e., $\beta|H| = |\{v' \in H' | R^{(v')} \equiv R\}|$. Clearly, for each $v'' \in H'$ there can be at most $\beta|H|$ directions $v' \in H'$ for which $R^{(v'')} \equiv R^{(v')}$. Hence, by Claim 2 it is

$$|H'| \cdot \beta \cdot |H| \geq \left((1-4\alpha)^2 - \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}}\right) \cdot |H|^2$$

and thus, using $|H'| \leq |H|$, we obtain

$$\beta \geq (1-4\alpha)^2 - \frac{2\alpha}{\frac{1}{3} - \frac{1}{40}\alpha - \frac{2hkd}{|F|}}. \tag{4.5}$$

From $\alpha := 10^{-2}$ in Definition 4.12 and our assumption that $|F| \geq 10^4(2hkd+1)^3$ it follows that $\beta > 0.84$.

Claim 4: The majority polynomial R and r_0 are different: $R \not\equiv r_0$.

Proof of Claim 4: Recall that by definition $r_0(t)$ equals $r_0^{(v')}(t)$ for each $v' \in H'$ and is the polynomial which is claimed by π on ℓ_{v^*,x^*}. Similarly, for the majority of $v' \in H'$ polynomial $R(t)$ equals $R^{(v')}(t)$. We prove Claim 4 by showing that the particular value $R(0)$ is attained for more choices of $v' \in H'$ than $r_0(0)$.

First note that item 4,ii) of Proposition 4.13 for all $v' \in H'$ implies $c_0^{(v')}(s) := p_{v^*,x^*+sv'}(\tau_{x^*+sv'}) = p_{v',x^*}(\tau_{x^*+sv'})$ for at least $\frac{1}{2}|F|$ values of s. Next, Lemma 4.14 implies for each $v' \in H'$ that for at least $\frac{2}{3}|F|$ values of s it holds $r_s^{(v')}(t) = Q^{(v')}(s,t)$ as polynomials in t. For those s it follows in particular that $Q^{(v')}(s,0) = r_s^{(v')}(0) = p_{v^*,x^*+sv'}(\tau_{x^*+sv'})$.

Combining the two equations results for each v' in at least $(\frac{2}{3}-\frac{1}{2})|F|$ many values for s for which $c_0^{(v')}(s) = Q^{(v')}(s,0)$. Now since both functions are univariate polynomials of degree at most khd they are equal as long as $|F|$ is large enough.

Next, it follows that $p_{v',x^*}(\tau_{x^*}) = c_0^{(v')}(0) = Q^{(v')}(0,0) = R^{(v')}(0)$; the latter by definition of $R^{(v')}$ and $p_{v',x^*}(\tau_{x^*})$ equals the value $R(0)$ for at least $\beta|H|$ choices of $v' \in H'$.

On the other hand it is $x^* \in S(v^*)$, thus for at most $\alpha|H|$ many $w \in H$ the value $r_0(0) = p_{v^*,x^*}(\tau_{x^*})$ coincides with $p_{w,x^*}(\tau_{x^*})$. But $\beta > \alpha$, therefore the claim $R \not\equiv r_0$ follows.

What remains to be done is to show that using R instead of r_0 in the corresponding segment of π strictly reduces its inconsistency.

Claim 5: The number of pairs (y,w) with $y \in \ell_{v^*,x^*}$ and $w \in H$ for which $p_{y,w}$ agrees with R on y is larger than the number of such pairs for which $p_{y,w}$ agrees with r_0 on y.

Proof of Claim 5: Since $R \neq r_0$ they agree in at most $2hkd$ points on ℓ_{v^*,x^*}. By the inclusion-exclusion principle it thus suffices to show that among the $|F||H|$ triples of form $(y, v^*, w), y \in \ell_{v^*,x^*}, w \in H$ (note that v^* is fixed) there are more than $(\frac{1}{2}|F| + 2hkd)|H|$ many for which $p_{w,y}$ agrees with R on x^*. By Lemma 4.14 and Claim 3 there exist $\beta|H|$ directions $v' \in H'$ for which the distance from $t \mapsto c_t^{(v')}(0)$ to R is at most $\frac{1}{3}$. It follows that

$$|\{(y,w) \in \ell_{v^*,x^*} \times H | p_{w,y} \text{ agrees with } R \text{ on } y\}| \geq \beta|H| \cdot \frac{2}{3}|F|.$$

Plugging in the bounds for β and $|F|$ gives $\beta|H| \cdot \frac{2}{3}|F| > (\frac{1}{2}|F| + 2hkd)|H|$. This finishes the proof of Claim 5 and thus also the one of Proposition 4.11. □

Theorem 4.6 now easily follows:

Proof. (of Theorem 4.6) We have shown that given a verification proof π and a function f which is a majority function of π and not a max-degree $2hkd$ polynomial we can construct a verification proof π' with a majority function f' such that the following holds.

- The univariate polynomials that π and π' claim differ on one line (i.e. π and π' differ in one segment) and f and f' disagree in at most $|F|$ places.
- The inconsistency of π' is strictly less than the inconsistency of π.

If we apply this construction iteratively it must come to an end after finitely many steps because the inconsistency cannot be reduced an unbounded number of times. Hence, at some point we must obtain a function f_s which is a max-degree $2hkd$ polynomial. Lemma 4.10 implies that for each (f_i, π_i) in the sequence the rejection probability is at most 2ϵ and this finishes the proof. □

4.3 Remaining proofs

This section contains all proofs lacking in the main text of the chapter.

4.3.1 Proof of Proposition 4.7

In this first subsection we fill in the remaining part to establish Theorem 4.5 by proving Proposition 4.7.

The intuitive importance of the proposition lies in the statement that the set H contains sufficiently many directions to cover F^k in a suitable way when used in the test. This is shown to be true by applying the theory of expander graphs[4]. All results on expanders used below can be found, for example, in [1].

Let G be the graph with elements of F^k as its vertices and an edge between two points $x, y \in F^k$ iff $x \neq y$ and there exists $v \in H$ such that $\ell_{v,x} = \ell_{v,y}$. Thus, two points are connected in G if they lie on a line having a direction in H. Since each point x lies on $|H|$ lines having direction in H and each such line contains $|F| - 1$ points different from x, the graph G is regular with degree $(|F| - 1)|H|$. Recall that Proposition 4.7 aims for getting many pairs $(x, y) \in S \times \bar{S}$ for a suitable S such that both points are connected by a line in H and the test realizes an error due to the definition of S. To obtain a sufficiently large detection probability there should be many edges connecting S and $\bar{S}$. This is precisely what expander graphs guarantee.

If we can show that G has a good expansion rate, then Proposition 4.7 directly follows from Theorem 2.22. By Definition 2.20 this means that we have to show that the largest eigenvalue in absolute value of the random walk matrix A of G is small. We have

$$A_{x,y} = \begin{cases} 0 & \text{if there is no edge connecting } x \text{ and } y \\ \frac{1}{(|F|-1)|H|} & \text{if there is an edge connecting } x \text{ and } y. \end{cases}$$

Here and below notationally we index rows and columns of A as well as components of (eigen)vectors in the obvious way by points $z \in F^k$. The main result of this subsection is the following.

Lemma 4.15. *The graph G has an expansion parameter $\lambda \leq \frac{1}{2h+1}$.*

The remainder of this subsection thus is devoted to the proof of Lemma 4.15. This is accomplished by computing explicitly all eigenvalues of A together with a corresponding basis of eigenvectors for F^k. Towards this aim a close correspondence between the latter and hyperplanes in F^k is established. The next lemma makes this more precise. Note that we are dealing with A as a real matrix and all computations below have to be done in $\mathbb{R}$, not in F.

[4]Though expander graphs play an important role in the proof of the PCP theorem given by Dinur below they enter in a completely different manner.

Lemma 4.16. *Let $L = \{x \in F^k | w^t \cdot x = c\}$ be a hyperplane in F^k, where $w \in F^k \setminus \{0\}$ and $c \in F$ are fixed. With L we associate a vector $V \in \mathbb{R}^{(F^k)}$ by defining its components $V_z, z \in F^k$ via*

$$V_z := \begin{cases} 1 - |F| & \text{if } z \in L \\ 1 & \text{if } z \notin L \end{cases}.$$

Let N be the number of $v \in H$ that are parallel to L, i.e., $N := |\{v \in H | w^t \cdot v = 0\}|$; note that N depends on w.

Then V is an eigenvector of A. The corresponding eigenvalue is $\frac{|F|N-|H|}{(|F|-1)|H|}$. Its absolute value is at most $\frac{N}{|H|}$ and thus always smaller than 1.

Proof. We show that $AV = \frac{|F|N-|H|}{(|F|-1)|H|}V$ by verifying the equation $(AV)_x = \frac{|F|N-|H|}{(|F|-1)|H|}V_x$ separately for each $x \in F^k$.

Suppose there are a columns indexed by an $z \in L$ and b columns indexed by an $z \notin L$ such that x is connected to z in G. Then

$$(AV)_x = \frac{a}{(|F|-1)|H|}(1-F) + \frac{b}{(|F|-1)|H|}.$$

We therefore have to count a and b in the different possible situations.

First suppose $x \in L$. Then it holds $a = N(|F|-1)$ and $b = (|H|-N)(|F|-1)$: In H there are N directions parallel to L and $|H|-N$ ones non-parallel. Each parallel $v \in H$ gives $|F|-1$ points on the line through x in direction v contributing to the value of a, similarly for non-parallel directions and b. Note here that those lines are disjoint (except for x) due to the definition of H. Thus, altogether it follows

$$(AV)_x = \frac{N(|F|-1)}{(|F|-1)|H|}(1-F) + \frac{(|H|-N)(|F|-1)}{(|F|-1)|H|} = \frac{N|F|-|H|}{(|F|-1)|H|}(1-F).$$

Next suppose that $x \notin L$. Then it is $a = |H|-N$ and $b = (|H|-N)(|F|-2)+N(|F|-1)$: The N lines through x in direction $v \in H$ which are parallel to L will not intersect L at all, so each contributes $|F|-1$ many points to b. The $|H|-N$ non-parallel lines intersect L in precisely one point (contributing to a) and do not intersect L in the remaining $|F|-2$ points (contributing to b). This again yields

$$(AV)_x = \frac{|H|-N}{(|F|-1)|H|}(1-F) + \frac{(|H|-N)(|F|-2)+N(|F|-1)}{(|F|-1)|H|} = \frac{N|F|-|H|}{(|F|-1)|H|}.$$

Finally, the stated estimations trivially hold because of $N < |H|$. □

As regular graph G always has eigenvalue 1. In order to obtain a bound on the expansion rate we have to give an upper bound on the second largest eigenvalue in absolute value. Towards this aim we next prove that the previous lemma gives all eigenvalues of A beside 1 together with a basis of eigenvectors; then an upper bound on the number N will suffice to get the expansion rate.

Choosing a hyperplane $L = \{x \in F^k | w^t \cdot x = c\}, w \in F^k \setminus \{0\}, c \in F$ there are $|F|^k - 1$ possibilities for w and $|F|$ choices for c. By normalizing $c \neq 0$ to $c = 1$ there are $(|F|^k - 1)|F|/(|F| - 1)$ hyperplanes in total and $|F|^k - 1$ many with $c \neq 0$. The following lemma shows that the vectors corresponding to these hyperplanes form a linearly independent set.

Lemma 4.17. *For $w \in F^k \setminus \{0\}$ let $V^{(w)}$ be the eigenvector of A corresponding to the hyperplane $\{x \in F^k | w^t \cdot x = 1\}$ according to Lemma 4.16. Then the set $\{V^{(w)} | w \in F^k \setminus \{0\}\}$ is linearly independent.*

Proof. Suppose there is a linear combination $\sum_{w \in F^k \setminus \{0\}} s_w \cdot V^{(w)} = 0$ for suitable $s_w \in \mathbb{R}$. For each fixed w we construct a vector which is orthogonal to all $V^{(w')}, w' \neq w$ but not to $V^{(w)}$. Scalar multiplication of the above equation with this vector yields $s_w = 0$ and thus the claim.

For fixed $w \in F^k, w \neq 0$ define vectors $L^{(w)}, Z^{(w)} \in \{0,1\}^{|F|^k}$ via their components:

$$L_x^{(w)} := \begin{cases} 1 & \text{if } w^t \cdot x = 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad Z_x^{(w)} := \begin{cases} 1 & \text{if } w^t \cdot x = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Note that in the defining conditions the scalar product still is understood over F^k, though the vectors are real. Thus, $L_x^{(w)} = 1$ iff $x \in L^{(w)}$ whereas $Z_x^{(w)} = 1$ iff x belongs to the plane parallel to $L^{(w)}$ passing through the origin.

This implies that $L^{(y)} - Z^{(y)}$ has the desired properties: Since each hyperplane contains $|F|^{k-1}$ many points it is

$$(V^{(w)})^t \cdot (L^{(w)} - Z^{(w)}) = (1 - |F|) \cdot |F|^{k-1} - |F|^{k-1} \neq 0.$$

For any $w' \neq w$ which is a multiple of w we have

$$V^{(w')} L^{(w)} = V^{(w')} Z^{(w)} = 1 \cdot |F|^{k-1}$$

and for any w' which is not a multiple of w it follows

$$V^{(w')} L^{(w)} = V^{(w')} Z^{(w)} = (|F| - 1) \cdot |F|^{k-2} + 1 \cdot (|F|^{k-1} - |F|^{k-2}).$$

In both cases the scalar product of $V^{(w')}$ and $L^{(w)} - Z^{(w)}$ vanishes and the claim follows. □

Finally, it remains to bound the eigenvalues that are smaller than 1 in absolute value. According to Lemma 4.16 it suffices to bound the number of directions in H parallel to a given plane.

Lemma 4.18. *For every hyperplane $L = \{x \in F^k | w^t \cdot x = c\}, w \in F^k \setminus \{0\}, c \in F$ the number N of directions $v \in H$ that are parallel to L is at most $\frac{1}{2h+1}|H|$.*

Proof. We show that for each $v \in H$ which is parallel to L there correspond at least $2h$ non-parallel directions in H. Moreover, the latter are all different for different choices of v. Consequently, N must satisfy $N \cdot (1 + 2h) \leq |H|$, thus yielding the lemma.

Suppose that $0 \neq v = (v_1, v_2, \ldots, v_k) \in H$ is parallel to L. Since $w \neq 0$ assume without loss of generality $w_1 \neq 0$. Not all $v_2, \ldots, v_k$ can vanish simultaneously because otherwise $w^t \cdot v = 0$ implies $v = 0$. Now consider the set

$$\{v' \in H | \exists t \in F \setminus \{0\} : \; v'_1 \neq v_1 \wedge tv'_2 = v_2 \wedge \cdots \wedge tv'_k = v_k\}.$$

It contains $2h$ directions in H that are not parallel to L. This is true because there are $2h$ choices for v'_1 different from v_1 that result in another element in H (note here that the factor t is only included for technical reasons due to our definition of H which excludes multiples of the same vector to belong to H). Given $w^t \cdot v = 0$ we must have $w^t \cdot v' \neq 0$ because of the changed first component.

Finally suppose there would be another $\tilde{v} \in H$ parallel to L and different from v such that the corresponding set of non-parallel directions would not be disjoint from that constructed for v. Then by definition of the set of non-parallel directions there is a $t \in F$ such that $t\tilde{v}_i = v_i$ for all $2 \leq i \leq k$, but $t \cdot \tilde{v}_1 \neq v_1$ because of definition of H. Since both scalar products $w^t \cdot v$ and $w^t \cdot \tilde{v}$ hinge on the first component they cannot be equal, so not both v and $\tilde{v}$ can be parallel to L. This contradicts the assumption. □

This finishes the proof of Lemma 4.15. Proposition 4.7 now follows from this and Theorem 2.22 by plugging in the values for d, λ, and $|S| = \delta|F|^k$.

4.3.2 Proof of Proposition 4.13

To show the existence of an $x^* \in F^k, v^* \in H$ and $H' \subseteq H$ as claimed in Proposition 4.13 a bunch of additional structural and combinatorial results related to the set H of test directions are needed. Let us first roughly describe the flavour of these results. Using our assumption that π has at most small inconsistency we show the existence of a large subset $H_1 \subseteq H$ of directions that contribute only little to this inconsistency. Moreover, any point in this set H_1 is nice with respect to the set S introduced in Definition 4.12. By this we mean that for all directions $v \in H_1$ the points in S are well distributed over lines with direction v. We then find a special direction $v^* \in H_1$ such that $S(v^*)$ has maximal cardinality among all $S(v), v \in H_1$ and $|S(v^*)| > |S|$. After this we show (using additional sets $H_2, \ldots, H_4$) the existence of a large subset $H_5 \subseteq H_1$ which is nice with respect to $S(v^*)$. The relatively large size of $S(v^*)$ and the properties of the directions in H_5 finally allow to show that most pairs $(x, v) \in F^k \times H_5$ satisfy the conditions in Proposition 4.13. From this the existence of an x^* and a large set $H' \subseteq H_5$ that meet the requirements of Proposition 4.13 finally can be concluded.

Though proofs are a bit tedious it is of course necessary to be convinced that H has the claimed properties. We therefore include all proofs in detail.

The first one shows that for a small set $T \subset F^k$ most lines with a direction in H that do contain points from T only contain a few of them.

Lemma 4.19. *Let $T \subseteq F^k$ be arbitrary with $T = \tau|F^k|$ for some $\tau \in (0,1]$. Let $\gamma > \frac{1}{|F|}$, then the fraction of pairs $(v,x) \in H \times T$ for which $\ell_{v,x}$ contains at least $\gamma|F|$ points from T is at most*

$$\frac{(1 - \frac{2h}{2h+1}(1-\tau))(|F|-1)}{\gamma|F|-1}.$$

Proof. The idea is to use Proposition 4.7 which holds for any arbitrary subset. Enumerate all lines in F^k with direction in H in an arbitrary way and denote by ℓ_i the number of points from T on line number i. The number of edges in graph G between T and its complement is

$$\sum_i \ell_i(|F| - \ell_i)$$

and from Proposition 4.7 it follows that

$$\sum_i \ell_i(|F| - \ell_i) \geq \frac{2h}{2h+1}(1-\tau)(|F|-1)|H|\tau|F|^k.$$

Since every point in F^k is incident with $|H|$ lines that have a direction in H it follows that $\sum_i \ell_i = \tau \cdot |H| \cdot |F|^k$. Thus the above inequality can also be rewritten as

$$\left(|F| - \frac{2h}{2h+1}(1-\tau)(|F|-1)\right)\sum_i \ell_i - \sum_i \ell_i^2 \geq 0.$$

Considering the above lines being represented by pairs $(v,x) \in H \times T$ it is easy to see that $\sum_{(v,x)\in H\times T} 1 = \sum_i \ell_i$ as well as $\sum_{(v,x)\in H\times T} |\ell_{v,x} \cap T| = \sum_i \ell_i^2$. The above inequality thus becomes

$$\sum_{(v,x)\in H\times T} \left(|F| - \frac{2h}{2h+1}(1-\tau)(|F|-1)\right) - |\ell_{v,x} \cap T| \geq 0. \tag{4.6}$$

In this sum a $(v,x) \in H\times T$ contributes at most $c_1 := \left(|F| - \frac{2h}{2h+1}(1-\tau)(|F|-1)\right) - 1$. This is the case if line $\ell_{v,x}$ only contains x as point in T. On the other hand, if a line $\ell_{v,x}$ contains at least $\gamma|F|$ points from T the pair (v,x) contributes at most $c_2 := \left(|F| - \frac{2h}{2h+1}(1-\tau)(|F|-1)\right) - \gamma|F|$. Now suppose a fraction θ of pairs in $H\times T$ contains at least $\gamma|F|$ points from T. Then enlarging the left-hand side from (4.6) gives $(1-\theta)c_1 + \theta c_2 \geq 0$ and thus

$$\theta \leq \frac{c_1}{c_1 - c_2} = \frac{c_1}{\gamma|F|-1}.$$

From this the lemma follows. □

Lemma 4.20. *Let $\gamma := \frac{1}{40}\alpha$ be fixed for the rest of the chapter. Let π be the verification proof from Proposition 4.13. Remember that π's inconsistency is $\leq 2\epsilon$ for $\epsilon \leq 10^{19}$.*

Define a subset $H_1 \subset H$ as those $v \in H$ which satisfy the following two conditions (see Definition 4.12):

i) $|S(v)| \leq \frac{16\epsilon}{\alpha(1-\alpha)} \cdot |F^k|$ *and*

ii) there are at most $\gamma^2 \cdot |S|$ points $x \in S$ such that the line $\ell_{v,x}$ contains at least $\gamma|F|$ points from S.

Then the cardinality of H_1 satisfies $|H_1| \geq (1 - \frac{1}{4}\alpha)|H|$.

Proof. For both conditions it will be shown that if a large fraction of directions $v \in H$ does not satisfy that condition, then the inconsistency of π must be larger than 2ϵ.

Let β_1 denote the fraction of directions $v \in H$ that do not satisfy the first condition, i.e., it is $|S(v)| > \frac{16\epsilon}{\alpha(1-\alpha)} \cdot |F^k|$. For each of this $\beta_1|H|$ many v by definition of $S(v)$ there are $\geq (1-\alpha)\frac{16\epsilon}{\alpha(1-\alpha)}|F^k|$ many pairs $(v', x) \in H \times F^k$ for which $p_{v',x}(\tau_x) \neq p_{v,x}(\tau_x)$. Each such triple (x, v, v') contributes to π's inconsistency, so we get at least an inconsistency of $\beta_1 \cdot \frac{16\epsilon}{\alpha}$. By assumption this must stay below at most 2ϵ, implying $\beta_1 \leq \frac{1}{8}\alpha$.

Next, an upper bound on the fraction β_2 of directions $v \in H$ that do not satisfy the second condition will be derived for the case that S is nonempty. If S is empty the second condition is trivially satisfied for all $v \in H$. In order to do so we first give an upper bound on $|S|$. For every $x \in S$ the fraction of pairs $(v', v'') \in H^2$ for which $p_{v',x}(\tau_x) \neq p_{v'',x}(\tau_x)$ is at least $2\alpha(1-\alpha)$. Again, each such triple (x, v', v'') contributes to π's inconsistency. It follows that $2\alpha(1-\alpha)\frac{|S|}{|F^k|} \leq 2\epsilon$ and thus

$$|S| \leq \frac{\epsilon}{\alpha(1-\alpha)} \cdot |F^k|.$$

Now suppose for $\beta_2|H|$ many directions $v \in H$ there is more than a γ^2 fraction of points $x \in S$ such that line $\ell_{v,x}$ contains at least $\gamma|F|$ points from S. This means that the fraction of pairs $(v, x) \in H \times S$ for which the line $\ell_{v,x}$ contains at least $\gamma|F|$ points from S is at least $\gamma^2\beta_2$. Choosing $\tau = \frac{|S|}{|F^k|} \leq \frac{\epsilon}{\alpha(1-\alpha)}$ and $T := S$ in Lemma 4.19 this fraction can be at most

$$\frac{(1 - \frac{2h}{2h+1}(1 - \frac{\epsilon}{\alpha(1-\alpha)}))(|F| - 1)}{\gamma|F| - 1}.$$

It follows that (using the values for h, α and γ and the bounds for ϵ and $|F|$)

$$\beta_2 \leq \frac{(1 - \frac{2h}{2h+1}(1 - \frac{\epsilon}{\alpha(1-\alpha)}))(|F| - 1)}{\gamma^2(\gamma|F| - 1)} \leq \frac{1}{8}\alpha.$$

Thus, both conditions are violated for at most $\frac{1}{8}\alpha|H|$ points, implying the cardinality statement for H_1 to be true. □

The previous lemma showed that for elements $v \in H_1$ the set $S(v)$ is not too large. In the next lemma we will show that vice versa there exists a $v \in H_1$ for which $S(v)$ as well is not of too small cardinality. For such a v there are relatively many $x \in F^k$ such that $p_{v,x}$ disagrees with many other line polynomials on x. This v will be the direction v^* whose existence is claimed in Proposition 4.13.

Lemma 4.21. *Let H_1 be as in Lemma 4.20 and π be as in Proposition 4.13. There exists $v^* \in H_1$ such that $|S(v^*)| \geq |S|$.*

Proof. The proof idea is as follows: First construct two directions $w, w' \in H_1$ for which there are at least $\alpha|S|$ points $x \in S$ with $p_{w,x}(\tau_x) \neq p_{w',x}(\tau_x)$. Then Lemma 4.14 is applied to the planes in F^k generated by (w, w'); using the properties of w and w' as elements in H_1 there must be as well many points $x \in F^k \setminus S$ for which $p_{w,x}(\tau_x) \neq p_{w',x}(\tau_x)$. Since each such point must either belong to $S(w)$ or to $S(w')$ one of those sets must be large.

Now towards the details.

Claim 1: There exist $w, w' \in H_1$ such that for at least $\alpha|S|$ points $x \in S$ it holds $p_{w,x}(\tau_x) \neq p_{w',x}(\tau_x)$.

Proof of Claim 1: For every $x \in S$ the number of pairs $(v', v'') \in H^2$ for which $p_{v',x}(\tau_x) \neq p_{v'',x}(\tau_x)$ is at least $2\alpha|H| \cdot (1-\alpha)|H|$. At most $(1-(1-\frac{1}{4}\alpha)^2)|H|^2$ of those pairs do not belong to H_1^2. Hence there are at least (using the fact that α is small enough)

$$(2\alpha(1-\alpha) - \frac{1}{2}\alpha + \frac{1}{16}\alpha^2)|H|^2 \geq \alpha|H|^2 \geq \alpha|H_1|^2$$

pairs $(v', v'') \in H_1^2$ for which $p_{v',x}(\tau_x) \neq p_{v'',x}(\tau_x)$. By the pigeonhole principle there must exist w, w' with the required properties.

Let us fix these w, w' and define $T \subset S$ as set of points $x \in S$ for which $p_{w,x}(\tau_x) \neq p_{w',x}(\tau_x)$. Clearly,

$$|T| \geq \alpha|S|. \tag{4.7}$$

Claim 2: There are at least $(1 - 4\gamma^2/\alpha)|T|$ points $x \in T$ such that

i) both lines $\ell_{w,x}$ and $\ell_{w',x}$ contain at most $\gamma|F|$ points from T and

ii) the (w, w')-plane through x contains at most $2\gamma|F|^2$ points from T.

Proof of Claim 2:

The second condition of Lemma 4.20 guarantees that there are at most $2\gamma^2|S|$ points $x \in T$ not satisfying the first condition. Inequality (4.7) implies $2\gamma^2|S| \leq 2\gamma^2/\alpha|T|$ and thus there are at least $(1 - 2\gamma^2/\alpha)|T|$ points $x \in T$ that satisfy the first condition.

Next we count how many of those points can be located in a (w, w')-plane through this point which has more than $2\gamma|F|^2$ points from T. Each (w, w')-plane through F^k contains at most $\gamma|F|^2$ points from T satisfying condition i). This is true because one can partition the plane into $|F|$ lines along direction w and each such line trivially has at most $\gamma|F|$ points in T with condition i). Therefore, if such a plane contains more than $2\gamma|F|^2$ points from T at least half of them violate condition i). Thus the total number of points from T

satisfying condition i) and being located in such a plane with many points from T is less than or equal to the number of points in T in such planes not satisfying condition i). But the latter number by the first argument above was shown to be at most $2\gamma^2/\alpha|T|$.

This implies that among the at least $(1-2\gamma^2/\alpha)|T|$ points from T satisfying condition i) at most $2\gamma^2/\alpha|T|$ do not satisfy condition ii). Claim 2 follows.

Claim 3: If $x \in T$, then at least one of the following conditions holds:

i) The (w, w')-plane through x contains at least $\frac{1}{4}|F|^2$ points from $S(w) \cup S(w') \cup T$.

ii) At least one of the lines $\ell_{w,x}$ or $\ell_{w',x}$ contains at least $\frac{2}{3}|F| - 2hkd$ points from $S(w) \cup S(w') \cup T$.

Proof of Claim 3: Assume that the first condition is not satisfied. Note that by definition of the corresponding sets if a point $x \notin S(w) \cup S(w') \cup T$, then $p_{w,x}(\tau_x) = p_{w',x}(\tau_x)$. Thus, assuming condition i) to be violated there are $\geq \frac{3}{4}|F|^2$ agreements along lines in directions w and w'. Let $x \in T$ and define univariate polynomials

$$\begin{aligned} r_s(t) &:= p_{w,x+sw'}(\tau_{x+tw+sw'}) \\ c_t(s) &:= p_{w',x+tw}(\tau_{x+tw+sw'}). \end{aligned}$$

Now apply Lemma 4.14. Hence, there exists a bivariate max-degree hkd polynomial Q such that for a $\frac{2}{3}$-fraction of choices $s \in F$ it is $r_s(t) \equiv Q(s,t)$; similarly, for a $\frac{2}{3}$-fraction of values $t \in F$ it is $c_t(s) \equiv Q(s,t)$.

So for $s \in F$ fixed, if $r_s(t) \not\equiv Q(s,t)$, then $r_s(t)$ equals $Q(s,t)$ in at most $2hkd$ values of t. Since for at most $\frac{1}{3}|F|$ values of t it can be the case that $c_t(s) \neq Q(s,t)$ it follows that $r_s(t)$ can agree with $c_t(s)$ in at most $\frac{1}{3}|F| + 2hkd$ values of t. In the same way for fixed $t \in F$, if $c_t(s) \not\equiv Q(s,t)$, then $c_t(s)$ agrees with $r_s(t)$ in at most $\frac{1}{3}|F| + 2hkd$ values of s.

Finally, $x \in T$ implies $r_0(0) = p_{w,x}(\tau_x) \neq p_{w',x}(\tau_x) = c_0(0)$. Thus, at least one of the two values $r_0(0)$ and $c_0(0)$ differs from $Q(0,0)$. Suppose this to be $r_0(0)$, then $r_0(t)$ agrees with $Q(0,t)$ in at most $2hkd$ values for t, whereas $c_t(0) = Q(0,t)$ for at least $\frac{2}{3}|F|$ values of t. It follows that $p_{w,x}(\tau_{x+tw}) = r_0(t) \neq c_t(0) = p_{w',x+tw}(\tau_{x+tw})$ for at least $\frac{2}{3}|F| - 2hkd$ values of t. Each such point $x + tw$ belongs either to T (if it belongs to S), to $S(w)$ or to $S(W')$. The same argument for the case that $c_t(0) \neq Q(0,t)$ finishes the proof of Claim 3.

Claim 4: Each (w, w')-plane that contains at most $2\gamma|F|^2$ points from T satisfies the following: The number of points in this plane belonging to $S(w) \cup S(w') \cup T$ is at least by a factor $\frac{1}{8\gamma}$ as large as the number of points in T that satisfy all the conditions of Claim 2.

Proof of Claim 4: Suppose that we have a (w, w')-plane which contains at most $2\gamma|F|^2$ points from T; so condition ii) of Claim 2 holds. Given $x \in T$ at least one of the two conditions of Claim 3 hold. Suppose the first condition of Claim 3 to be true. Then the plain contains $\frac{1}{4}|F|^2$ points from $S(w) \cup S(w') \cup T$ and thus the number of points from $S(w) \cup S(w') \cup T$ on this plane is at least $\frac{1}{8\gamma}$ times as large as the number of points from T on this plane. If the first condition of Claim 3 is violated the second must hold for each point

x of the plane belonging to T. Thus at least one of the lines $\ell_{w,x}$ or $\ell_{w'x}$ contains at least $\frac{2}{3}|F|-2hkd$ points from $S(w)\cup S(w')\cup T$. Using this to count points from $S(w)\cup S(w')\cup T$ for each $x \in T$ which satisfies the conditions of Claim 2 might results in counting them multiple times. However, since by Claim 2 at most $\gamma|F|$ points from T are located on each of the lines no point is counted more than $2\gamma|F|$ times this way. Therefore, in this case there are at least $\frac{1}{2\gamma}(\frac{2}{3} - \frac{2hkd}{|F|})$ times as many points in $S(w) \cup S(w') \cup T$ located on the (w, w')-plane than points from T that satisfy all conditions of Claim 2. Since $\frac{1}{8\gamma} < \frac{1}{2\gamma}(\frac{2}{3} - \frac{2hkd}{|F|})$ Claim 4 follows.

The proof of the lemma now easily follows. Choosing $\gamma \leq \frac{1}{40}$ it is

$$|S(w) \cup S(w') \cup T| \geq \frac{1}{8\gamma}(1 - 4\gamma^2/\alpha)|T| = \frac{1}{8\gamma}(1 - 4\gamma^2/\alpha)\alpha|S| \geq 5(1 - 4\gamma^2/\alpha)|S| \geq 4|S|.$$

Since $|T| < |S|$ it finally follows $|S(w) \cup S(w')| \geq 3|S|$ and thus at least one of the two sets has cardinality at least $|S|$. The corresponding direction now determines the v^* in the lemma's statement. □

Let $v^* \in H_1$ denote a point such that $|S(v^*)|$ is maximal. By Lemma 4.20 we have $|S(v^*)| \leq \frac{16\epsilon}{\alpha(1-\alpha)}|F|^k$ and by Lemma 4.21 it is $|S(v^*)| \geq |S|$. The next lemma will imply that $S(v^*)$ is non-empty.

Lemma 4.22. *Let π be a list of univariate line polynomials with direction in H as usual and let $f : F^k \to \mathbb{R}$ be the majority function defined by π. Suppose there is a subset $H_2 \subseteq H$ satisfying $|H_2| \geq \frac{7}{8}|H|$ such that for all $v, v' \in H_2$ and all $x \in F^k$ it holds $p_{v,x}(\tau_x) = p_{v',x}(\tau_x)$. Then f is a max-degree $2hkd$ polynomial.*

Proof. First note that with the assumptions f is uniquely defined on F^k; for each $x \in F^k$ at least $\frac{7}{8}$ of the values claimed for x by π are equal.

Let $\{e_i, 1 \leq i \leq k\}$ be the set of unit vectors in F^k. It is not hard to see that f is a multivariate max-degree $2hkd$ polynomial if for every $x \in F^k$ and all $1 \leq i \leq k$ the univariate restrictions $t \mapsto f(x+te_i)$ are polynomials of degree at most $2hkd$. This follows similarly as the analogue statement for algebraic polynomials, see for example [4]. Thus the goal is to verify the latter condition. Nevertheless, there is one technical difficulty. The set H_2 does not necessarily contain all unit vectors. This problem is circumvented as follows. It will be shown that for all e_i there exists $v, v' \in H_2$ such that at least one of the four vectors $\pm v \pm v'$ gives e_i. Suppose for example that $e_i = v + v'$. Consider the plane spanned by v, v' and an arbitrary point $x \in F^k$ in it. By assumption, f restricted to any line in this plane having direction v is a degree hkd polynomial, and similarly for direction v'. Thus, by Lemma 4.14 the bivariate function $(s, t) \mapsto f(x + sv + tv')$ is a max-degree hkd polynomial and therefore its restriction $q \mapsto f(x + qe_i) = f(x + q(v + v'))$ to line e_i through x is a degree $2hkd$ polynomial. Similarly for the other cases to represent e_i. The proof therefore is finished once we have shown that all unit vectors can be obtained that way.

Towards this aim let $H_3 \supseteq H$ which contains for every $v \in F^k$ with $|v| \leq h$ either v or $-v$; recall here Definition 4.4 of $|v|$. It is $|H_3| = \frac{1}{2}((2h+1)^k - 1)$. We now claim two things:

Claim 1: Any subset of H_3 containing at least $\frac{2}{3}$ of the elements of H_3 contains for each unit vector e_i a pair of vectors v, v' such that at least one among the four vectors $\pm v \pm v'$ equals e_i.

Here we use the shorthand notation $e_i = \pm v \pm v'$ to express that e_i satisfies at least one of these equations.

Claim 2: $|H| \geq \frac{7}{8}|H_3|$

Both claims together prove the result because starting from a set $H_2 \subseteq H$ with $|H_2| \geq \frac{7}{8}|H|$ it follows by Claim 2 that $|H_2| \geq \frac{49}{64}|H_3| > \frac{2}{3}|H_3|$; now Claim 1 implies the lemma.

Proof of Claim 1: Assume $H_4 \subseteq H_3$ satisfies $|H_4| > \frac{2}{3}|H_3|$ and fix an arbitrary $i \leq k$. For every $v \in H_3$ there exists at least one and at most two $v' \in H_3$ such that $e_i = \pm v \pm v'$. This is true because v' is a solution if $v' = \pm(e_i + v)$ or $v' = \pm(e_i - v)$; now at most two of the corresponding four right-hand side vectors belong to H_3, but depending on v only at least one additionally satisfies that its components remain $\leq h$ in absolute value.

Now build a graph G with the points in H_3 as vertices and an edge between two v, v' iff $e_i = \pm v \pm v'$. The previous argument implies that each vertex is incident with at least one and at most two edges. Claim 1 follows if we show that the set H_4 induces a subgraph with at least one edge. Suppose this is not the case. Then there is a subset U of less than $\frac{1}{3}|H_3|$ vertices such that every edge in G is incident with one of the vertices in U. Since no vertex of G is incident with more than two edges U covers less than $\frac{2}{3}|H_3|$ edges. But more than $\frac{2}{3}|H|$ vertices do not belong to U, so if each of them would be adjacent only to vertices in U more than $\frac{2}{3}|H|$ edges are necessary. Contradiction.

Proof of Claim 2: Let v be a random element in H_3. The probability that v does not contain an entry ± 1 is $\left(\frac{2h-1}{2h+1}\right)^k$; similarly for the probability that v does not contain an entry $\pm h$. Hence, the probability that v contains both an entry ± 1 and an entry $\pm h$ is at least $1 - 2\left(\frac{2h-1}{2h+1}\right)^k$. If v contains both an entry ± 1 and an entry $\pm h$, then multiplying v by any factor other than ± 1 will result in a point outside H_3 (note here that $|F| \geq 2h^2$).

For such a v either v or $-v$ belongs to H, so

$$|H| \geq \left(1 - 2\left(\frac{2h-1}{2h+1}\right)^k\right)|H_3|.$$

Finally,choosing $k \geq \frac{3}{2}(2h+1)$ yields

$$\left(\frac{2h-1}{2h+1}\right)^k \leq \frac{1}{16}$$

and thus the claim. □

This now implies that $S(v^*)$ is non-empty. Assume for a contradiction that $S(v^*)$ is empty. Then for all $v \in H_1$, $S(v)$ is empty and S is also empty. Hence, for all $v, v' \in H_1$

and for all $x \in F^k$ it holds $p_{v,x}(t_x) = p_{v',x}(t_x)$. Since $|H_1| \geq \frac{7}{8}|H|$ according to Lemma 4.20 the above lemma yields that the majority function already would be a max-degree $2hkd$ polynomial, contradicting our assumption that this is not the case.

We are finally able to finish the

Proof of Proposition 4.13:

Once more, the arguments are split into a few small claims.

Claim 1: There are at most $\gamma|S(v^*)|$ points $x \in S(v^*)$ such that the line $\ell_{v^*,x}$ contains more than $\gamma|F|$ points from S.

Proof of Claim 1: We show that there are at most $\gamma|S|$ points $x \in F^k$ such that the line $\ell_{v^*,x}$ contains more than $\gamma|F|$ points from S. Since $|S| \leq |S(v^*)|$ the claim follows. Towards this aim it will suffice to show that there are at most $\frac{\gamma|S|}{|F|}$ lines with direction v^* that contain at least $\gamma|F|$ points from S. By Lemma 4.20 there are at most $\gamma^2|S|$ points in S that lie on such a line. Every such line contains at least $\gamma|F|$ points from S, so it follows that there can be at most $\frac{\gamma^2|S|}{\gamma|F|} = \frac{\gamma|S|}{|F|}$ such lines.

For the next claims we define yet another subset $H_5 \subseteq H_1$ as all $v \in H_1$ such that there is at most a γ^2 fraction of points $x \in S(v^*)$ for which line $\ell_{v,x}$ contains at least $\gamma|F|$ points from $S(v^*)$ (so points in H_5 satisfy condition ii) of Lemma 4.20 with $S(v^*)$ instead of S).

Claim 2: For every $v \in H_5$ there are at least $(1 - \gamma^2 - 40\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that at most a $(\gamma + \frac{1}{10})$ fraction of points on $\ell_{v,x}$ belong to $S(v^*) \cup S(v) \cup S$.

Proof of Claim 2: Consider an arbitrary direction v in H_5 and let a denote the number of points $x \in S(v^*)$ for which $|\ell_{v,x} \cap S(v^*)| \leq 20\gamma|\ell_{v,x} \cap S(v)|$. Intuitively, a is the number of points in $S(v^*)$ that lie on a line in direction v on which the number of points from $S(v)$ is much bigger than the number of points from $S(v^*)$. By definition of v^* it is $|S(v^*)| \geq |S(v)|$, therefore it must be the case that a is relatively small. More precisely, it holds $|S(v)| \geq \frac{a}{20\gamma}$ or equivalently $a \leq 20\gamma|S(v)| \leq 20\gamma|S(v^*)|$. By the same argument using Lemma 4.21 there are at most $20\gamma|S| \leq 20\gamma|S(v^*)|$ points $x \in S(v^*)$ for which $|\ell_{v,x} \cap S(v^*)| \leq 20\gamma|\ell_{v,x} \cap S|$.

Next, $v \in H_5$ so by definition of H_5 there is a fraction of at most γ^2 points x of $S(v^*)$ for which $|\ell_{v,x} \cap S(v^*)| \geq \gamma|F|$.

The complement of those point sets counted in the arguments above are those at least $(1-\gamma^2-40\gamma)|S(v^*)|$ many points x in $S(v^*)$ for which both the line $\ell_{v,x}$ contains at most $\gamma|F|$ points from $S(v^*)$, $|\ell_{v,x} \cap S(v^*)| > 20\gamma|\ell_{v,x} \cap S(v)|$, and $|\ell_{v,x} \cap S(v^*)| > 20\gamma|\ell_{v,x} \cap S(v)|$.

It follows that the number of points that such a line contain which additionally belong to $S(v^*) \cup S(v) \cup S$ is bounded from above by $(\gamma + \frac{1}{20\gamma}\gamma + \frac{1}{20\gamma}\gamma)|F| = (\gamma + \frac{1}{10})|F|$.

Note that this fraction at most $\frac{1}{2}$.

Claim 3: For every $v \in H_5$ there are at least $(1 - 2\gamma^2 - 40\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that at most a $(2\gamma + \frac{1}{5})$-fraction of points in the (v, v^*)-plane through x belong to $S(v^*) \cup S(v) \cup S$.

Proof of Claim 3: Let $v \in H_5$ be arbitrarily chosen and fixed. As done several times before partition F^k into (v^*, v)-planes. We show that only a few points from $S(v^*)$ lie on

such a plane which contains at least $2\gamma|F|^2$ points from $S(v^*)$. Consider such a plane P. Then by the pigeonhole principle at least $\gamma|F|^2$ among the points in $P \cap S(v^*)$ are located on lines with direction v which contain $\geq \gamma|F|$ points from $S(v^*)$ each. Once again, by definition of H_5 there are at most $\gamma^2|S(v^*)|$ points in $S(v^*)$ lying on a line with direction v and having more than $\gamma|F|$ points from $S(v^*)$. Comparing the two bounds yields that at most $2\gamma^2|S(v^*)|$ points in $S(v^*)$ lie on a (v^*, v)-plane containing at least $2\gamma|F|^2$ points from $S(v^*)$.

In the same way as done for Claim 2 there are at most $20\gamma|S(v^*)|$ points $x \in S(v^*)$ which lie on a plane on which the number of points from $S(v^*)$ is at most 20γ times the number of points from $S(v)$. Similarly, there are at most $20\gamma|S(v^*)|$ points $x \in S(v^*)$ lying on a plane on which the number of points from $S(v^*)$ is at most 20γ times the number of points from S.

Altogether, we obtain at least $(1 - 2\gamma^2 - 40\gamma) \cdot |S(v^*)|$ points $x \in S(v^*)$ lying on a plane that contains less than $2\gamma|F|^2$ points from $S(v^*)$, less than $\frac{1}{20\gamma}2\gamma|F|^2$ points from $S(v)$ and less than $\frac{1}{20\gamma}2\gamma|F|^2$ points from S. This implies that there are at least $(1-2\gamma^2-40\gamma)\cdot|S(v^*)|$ points in $x \in S(v^*)$ that lie on a plane on which at most a fraction of $2\gamma + \frac{4\gamma}{20\gamma} = 2\gamma + \frac{1}{5}$ of points belong to $S(v^*) \cup S(v) \cup S$. This ends the proof of Claim 3.

All we need to do now to finish the proof of Proposition 4.13 is to show that there exists $x^* \in S(v^*)$ and $H' \subset H_5$ with $|H'| \geq (1-4\alpha)|H|$ such that conditions 2 and 4 are satisfied.

Let $v \in H_5$ be an arbitrary direction.
Claim 1 shows that there are at least $(1-\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that at most $\frac{1}{40}\alpha \cdot |F|$ points on $\ell_{v^*,x}$ belong to S (cf. item 2 of Proposition 4.13).
Claim 2 shows that there are at least $(1-\gamma^2-40\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that the fraction of $s \in F$ for which $p_{v^*,x+sv}(\tau_{x+sv}) \neq p_{v,x}(\tau_{x+sv})$ is at most $\frac{1}{2}$ (cf. item 4,ii) of Proposition 4.13).
Claim 3 shows that there are at least $(1-2\gamma^2-40\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that the fraction of $(t,s) \in F^2$ for which $p_{v^*,x+sv}(\tau_{x+tv^*+sv}) \neq p_{v,x+tv^*}(\tau_{x+tv^*+sv})$ is at most $\frac{1}{4}$. (c.f. item 4,i) of proposition 4.13).

Putting this together it follows that for every $v \in H_5$ there are at least $(1-3\gamma^2-81\gamma)|S(v^*)|$ points $x \in S(v^*)$ such that the pair (v,x) satisfies conditions 2 and 4 of Proposition 4.13. By the pigeon hole principle there exists $x^* \in S(v^*)$ and $H' \subset H_5$ with $|H'| \geq (1-3\gamma^2-81\gamma)|H_5|$ such that conditions 2 and 4 of Proposition 4.13 are satisfied by x^*, v^* and H'.

It remains to show that $(1-3\gamma^2-81\gamma)|H_5| \geq (1-4\alpha)|H|$. We can calculate an upper bound for the number of directions $v \in H_1$ that do not belong to H_5 in the same way that we calculated an upper bound on the number of directions $v \in H$ that do not satisfy condition ii) in Lemma 4.20. The only difference is that here we have to take $\tau = \frac{|S(v^*)|}{|F^k|} \leq \frac{16\epsilon}{\alpha(1-\alpha)}$. This leads to an upper bound of

$$\frac{(1-\frac{2h}{2h+1}(1-\frac{16\epsilon}{\alpha(1-\alpha)}))(|F|-1)}{\gamma^2(\gamma|F|-1)}|H| \leq \frac{1}{8}\alpha|H|$$

for the number of directions v that belong to H_1 but not to H_5. Hence $|H_5| \geq \frac{5}{8}\alpha|H|$ and thus $|H'| \geq (1-3\gamma^2-81\gamma)(1-\frac{3}{8}\alpha)|H| \geq (1-3\alpha)(1-\alpha)|H| \geq (1-4\alpha)|H|$ which is what we needed to show. □

4.3.3 Proof of Lemma 4.14.

The proof for trigonometric polynomials with minor changes follows the one of Lemma 5.2.1 in [3] for algebraic polynomials; it is included only for sake of completeness.

Consider two families of degree d polynomials $\{r_s\}_{s\in F}$ and $\{c_t\}_{t\in F}$, where $t \mapsto r_s(t)$ claims to describe row s of an $|F| \times |F|$ matrix and $s \mapsto c_t(s)$ claims to describe column t of this matrix. If $r_s(t) \neq c_t(s)$ for a pair (s,t) we say that there is a *disagreement*.

Choose $10 \cdot (2d+1)$ different values $t_1, \ldots, t_{10(2d+1)} \in F$ which as usual are used to number columns and such that in each of these columns at most a $\frac{5}{18}$-fraction of disagreements occur. Such columns exist because otherwise the assumption on the number of disagreements yields $(1 - \frac{10(2d+1)}{|F|}) \cdot \frac{5}{18} \leq \frac{1}{4}$, which in turn implies a contradiction to $|F| \geq 10^4(2d+1)^3$.

We shall in several steps show that there exists a submatrix consisting of $6 \cdot (2d+1)$ columns out of $t_1, \ldots, t_{10(2d+1)}$ and a $\frac{21}{72}$ fraction of rows that do not contain any disagreement. This will finally suffice to define the bivariate polynomial Q.

Definition 4.23. *Let $t_i \in F, 1 \leq i \leq 10(2d+1)$ be chosen as above. Consider an interpolation problem $\{(t_i, p_i) | p_i \in \mathbb{R}, 1 \leq i \leq 10(2d+1)\}$ for trigonometric polynomials of degree d. We call the data* well described *if there exists a trigonometric degree d polynomial r such that $r(t_i) = p_i$ for at least $6(2d+1)$ values of i.*

In case of existence the polynomial r in the above definition is unique because any other polynomial with the same property would have to agree with it in at least $2(2d+1)$ points.

Lemma 4.24. *Let $\{(t_i, p_i) | p_i \in \mathbb{R}, 1 \leq i \leq 10(2d+1)\}$ be well described.*

a) There is a polynomial b of degree $d+2(2d+1)$ and a nonzero polynomial e of degree $2(2d+1)$ such that

$$b(t_i) = e(t_i) \cdot p_i \quad \text{for all } i = 1, 2, \ldots, 10(2d+1). \tag{4.8}$$

b) For any two polynomials b, e satisfying equation (5.2) it holds that e divides b and b/e is the *polynomial that well describes the data set $\{(t_1, p_1), \ldots, (t_{10(2d+1)}, t_{10(2d+1)})\}$.*

Proof. Let r be the degree d polynomial that well describes the data set.

a) Define e as a degree $2(2d+1)$ polynomial such that $e(t_i) = 0$ for all i for which $r(t_i) \neq p_i$. Since the latter equality is violated for at most $4(2d+1)$ many t_i, the resulting interpolation problem for e allows to require one more value. So this final one is chosen to guarantee that e does not vanish identically. Now define $b(t_i) := r(t_i) \cdot e(t_i)$; then it follows $b(t_i) := p_i \cdot e(t_i)$ for all $i \leq 10(2d+1)$ proving part a).

b) Assume that e and b are any polynomials of appropriate degree such that $b(t_i) = p_i \cdot e(t_i)$ for all $i \leq 10(2d+1)$. Since r well describes the set $\{(t_1, p_1), \ldots, (t_{10(2d+1)}, p_{10(2d+1)})\}$ it follows that for $6(2d+1)$ values of i the equation $b(t_i) = r(t_i) \cdot e(t_i)$ holds. Since $b(t)$ and $r(t) \cdot e(t)$ are degree $d + 2(2d+1)$ polynomials and agree at $6(2d+1)$ arguments they must be equal. Hence we have $b/e = r$ as trigonometric polynomials. □

Call a row *good* if the number of disagreements among the columns $t_1, \ldots, t_{10(2d+1)}$ is at most $4(2d+1)$. Since in each of those columns t_i at most a $\frac{5}{18}$-fraction of disagreements occur, the total number of disagreements in those columns is bounded by $\frac{50}{18}(2d+1)|F|$. This implies that there are at least $\frac{11}{36}|F|$ good rows: otherwise the bad rows would contribute too many disagreements. In a good row s the data set $\{(t_1, c_{t_1}(s)), \ldots, (t_{10(2d+1)}, c_{t_{10(2d+1)}}(s))\}$ is well described by polynomial r_s. Hence, by Lemma 4.24 for every good row s there exist nonzero polynomials $b^{(s)}$ and $e^{(s)}$ of degree $d + 2(2d+1)$ and $2(2d+1)$, respectively, such that

$$b^{(s)}(t_i) = c_{t_i}(s) \cdot e^{(s)}(t_i) \quad \text{for every } i \leq 10(2d+1). \tag{4.9}$$

Our goal is to show that the polynomials participating in this equation can be seen as bivariate trigonometric polynomials in s, t and that it can be extended to hold for all values $s \in F$.

First note that writing $b^{(s)}(t) := b_0^{(s)} + \sum_{j=1}^{5d+2} b_j^{(s)} \cdot \cos(2\pi j \frac{t}{q}) + b'^{(s)}_j \cdot \sin(2\pi j \frac{t}{q})$ as well as $e^{(s)}(t) := e_0^{(s)} + \sum_{j=1}^{4d+2} e_j^{(s)} \cdot \cos(2\pi j \frac{t}{q}) + e'^{(s)}_j \cdot \sin(2\pi j \frac{t}{q})$ equation (5.3) becomes an overdetermined homogeneous linear system in the coefficients $b_j^{(s)}, b'^{(s)}_j, e_j^{(s)}, e'^{(s)}_j$ with $20d+10$ equations in $18d+10$ variables. Its coefficient matrix A has as entries real numbers as well as for varying s the trigonometric (in s) polynomials $c_{t_i}(s)$ of degree at most d.

Claim 1: Equation (5.3) is solvable for all choices of s in the space of trigonometric polynomials in s. The solution coefficients are polynomials of degree at most $5(2d+1)^2$.

Proof of Claim 1: For every good row s the system has a non-trivial solution. Thus, fixing such an s each $(18d+10) \times (18d+10)$ submatrix of A has vanishing determinant. Recall that the entries of A are polynomials in s of degree at most d. Thus as trigonometric polynomials in s all those determinants have degree bounded by

$$d \cdot (18d+10) < 10d(2d+1) \leq 5(2d+1)^2.$$

There are at least $\frac{11}{36}|F|$ good rows; our choice of F guarantees that this quantity is larger than $2 \cdot 5(2d+1)^2 + 1$. This implies that each above mentioned determinant vanishes identically. Consequently, equation (5.3) is non-trivially solvable for all $s \in F$, not only for those corresponding to good rows. Applying Cramer's rule (compare Fact A6 in [3]) there are solution coefficients $b_j^{(s)}, b'^{(s)}_j, e_j^{(s)}, e'^{(s)}_j$ being trigonometric polynomials is s of degree at most $d(18d+5) \leq 5(2d+1)^2$. They solve (5.3) for all s.

Writing down explicitly equation (5.3) using the solution coefficients it follows that equation (5.3) can be rewritten as

$$b(t_i, s) = c_{t_i}(s) \cdot e(t_i, s) \quad \text{for all } s \in F \text{ and } i \leq 10(2d+1),$$

where b and e now are bivariate trigonometric polynomials with t and s as variables. Their degrees with respect to t are $d + 2(2d+1)$ for b and $2(2d+1)$ for e, respectively. Both have degree $5(2d+1)^2$ in s.

Now e is a nonzero polynomial of degree at most $2(2d+1)$ in its first variable t; therefore it can be identically zero on at most $4(2d+1)$ columns. Assume without loss of generality that e is not identically zero on the columns $t_1, \ldots, t_{6(2d+1)}$.

Call a row s *nice* if $e(t_i, s) \neq 0$ for all $i \leq 6(2d+1)$.

Claim 2: There are at least $\frac{21}{72}|F|$ rows which are both good and nice. For such a row s it is $r_s(t_i) = c_{t_i}(s)$ for all $1 \leq i \leq 6(2d+1)$, i.e., here no disagreements occur.

Proof of Claim 2: With respect to variable s polynomial e has degree at most $5(2d+1)^2$. If e is not vanishing identically in a fixed column t_i it can have at most $10(2d+1)^2$ zeros. So a $1 - \frac{10(2d+1)^2 \cdot 6(2d+1)}{|F|} \geq 1 - \frac{1}{72}$ fraction of the rows have to be nice. For a nice row s it holds $c_{t_i}(s) = b(t_i, s)/e(t_i, s)$ for $i \leq 6(2d+1)$ and for a good row s by part b) of Lemma 4.24 we have $r_s(t_i) = b(t_i, s)/e(t_i, s)$ as polynomials when considering s as fixed. Thus in rows s that are both nice and good $r_s(a_i) = c_{a_i}(s)$ for all $i \leq 6(2d+1)$. By the above argument at least $(\frac{11}{36} - \frac{1}{72})|F| = \frac{21}{72}|F|$ rows have both properties.

For finishing the proof of Lemma 4.14 it remains to define the bivariate polynomial Q. Fix the $2d+1$ columns $t_i, 1 \leq i \leq 2d+1$ (the argument below works for any choice of $2d+1$ columns among the first $6(2d+1)$). Now let $Q(s,t) := \sum_{i=1}^{2d+1} L_{a_i}(t) \cdot c_{t_i}(s)$, where L_{a_i} is the degree d polynomial that is 1 at t_i and 0 at $\{t_1, \ldots, t_{2d+1}\} \setminus \{t_i\}$. Then Q is a max-degree d trigonometric polynomial in s, t.

For fixed s the $2d+1$ many values $Q(s, t_i), 1 \leq i \leq 2d+1$ determine the polynomial $t \mapsto Q(s,t)$. If s is one of the at least $\frac{21}{72}|F|$ many rows that are both good and nice Claim 2 implies that $Q(s, t_i) = c_{t_i}(s) = r_s(t_i)$, thus $t \mapsto Q(s,t)$ as well equals $r_s(t)$; moreover, $c_t(s) = r_s(t)$ then holds for all $t \in \{t_1, \ldots, t_{6(2d+1)}\}$.

Next recall that for any column $t_i, i \leq 10(2d+1)$ there are at most $\frac{5}{18}|F|$ disagreements $c_{t_i}(s) \neq r_s(t_i)$ with respect to varying s. But by the previous argument for each of these columns t_i there are at least $(\frac{21}{72} - \frac{5}{18})|F| \geq \frac{1}{72}|F|$ values for s such that $Q(s, t_i) = r_s(t_i) = c_{t_i}(s)$. Since $\frac{1}{72}|F| \geq 2d+1$ in fact the latter must hold for all values of s.

Finally, if $Q(s,t)$ disagrees with $r_s(t)$ for one of the other rows s, then there must be at least $9(2d+1)$ disagreements within the columns $t_i, 1 \leq i \leq 10(2d+1)$. But each such column by its definition can only contribute $\frac{5}{18}|F|$ disagreements. It follows that the number of rows for which $Q(s,t) \not\equiv r_s(t)$ is bounded by $\frac{5 \cdot 10}{9 \cdot 18}|F| \leq \frac{1}{3}|F|$ as was claimed.

Analogously, the statement can be obtained for the columns thus finishing the proof of the lemma. $\square$.

Chapter 5

Real PCP Theorem I: An Algebraic Proof

In this chapter we will give an algebraic proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem which has the same global structure as the Arora et al. proof but requires a lot of major modifications for it to work in the BSS setting over $\mathbb{R}$ and $\mathbb{C}$. The combinatorial proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem which follows Dinur's proof of the PCP theorem is less complicated and follows the classical proof more closely.

Let us start by giving an outline of the Arora et al. proof and point out where difficulties occur when trying to turn this proof into a proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem.

5.1 Problems with the classical proof and outline

The Arora et al. proof is built up as follows. Using truth tables of corresponding linear functions as codewords for strings of length n a linearity test is designed to construct a $(\mathrm{poly}(n), 1)$-restricted verifier for NP. This is what we did for the BSS model in Chapter 3. We shall use it below.

Next, a $(\log(n), \mathrm{polylog}(n))$-restricted verifier for NP is designed. It uses as codewords low-degree algebraic polynomials over suitable finite fields and designs a low-degree test together with a sum-check procedure to verify whether a codeword represents a satisfying assignment of a 3-SAT formula through a low-degree polynomial. Such verifiers have been shown to exist for $\mathrm{NP}_{\mathbb{R}}$ as well [36], but their structure is not appropriate to be used further on in the proof.

The low-degree test is not only important for itself, but used in [2] to put arbitrary verifiers into a more structured segmented form. Here, segmentation refers to the question how the verifier inspects the proof. It means that instead of arbitrarily exploiting the query resources it has, the verifier asks the information from the proof in a highly structured form by querying a constant number of blocks of length $\mathrm{polylog}(n)$. With this technique the $(\log(n), \mathrm{polylog}(n))$-restricted verifier is turned into an equivalent segmented verifier which makes it suitable for being used in a procedure called verifier composition. During this

process, the above verifier first is composed with itself once and then with the $(\text{poly}(n), 1)$-restricted verifier. The outcome of these two compositions is the required $(\log(n), 1)$-restricted verifier for NP.

It is here where further major difficulties arise when dealing with the real number setting. First, we need a low-degree test which at the same time respects the $(\log(n), \text{polylog}(n))$ resources and can be used to achieve a kind of segmentation procedure for other verifiers. The test we constructed in Chapter 4 meets these requirements. However, because of the use of trigonometric polynomials the test lacks the kind of structure that allows to prove a full segmentation lemma like the one in [2]. The difficulties for turning any real verifier into a segmented one with the low-degree test arise because the composition of a trigonometric polynomial with an algebraic polynomial is not a (low-degree) polynomial anymore. We solve this problem by proving a weak segmentation lemma. With this lemma at hand, we develop a variant of a sum-check procedure that can be segmented. So even if we are not able to give a full segmentation for any real verifier, our techniques solve the problem for a special sum-check sufficient in our setting. This finally clears the way to apply a real version of verifier composition and to obtain the $\text{PCP}_{\mathbb{R}}$ theorem.

This chapter is structured as follows. In Section 5.2 we again state the low-degree test from the previous chapter so that the reader does not have to look back all the time. Sections 5.3 and 5.4 are the main parts in this chapter. The main task to solve is the design of a segmented almost transparent proof for all problems in $\text{NP}_{\mathbb{R}}$. We show how to extend the low-degree test to check correctness of a given function in a fixed point. This is necessary for proving a weak segmentation lemma. The lemma then is used in order to design a sum-check test in segmented form. Both results are combined to obtain a segmented and $(\log(n), \text{polylog}(n))$-restricted verifier for $\text{NP}_{\mathbb{R}}$. These results in Section 5.5 finally enter a verifier composition procedure which is similar to the discrete one. Segmentation makes it working and leads to the full $\text{PCP}_{\mathbb{R}}$ theorem.

5.2 Testing trigonometric polynomials

As mentioned above the construction of a well structured $(\log(n), \text{polylog}(n))$-restricted verifier for $\text{NP}_{\mathbb{R}}$ is based on using trigonometric polynomials as code words for potential zeros of a polynomial system and a test procedure for such polynomials. For our definition of trigonometric polynomials and the motivation for using them, see Section 4.1

As starting point for the construction of a $(\log(n), \text{polylog}(n))$-restricted verifier for $\text{NP}_{\mathbb{R}}$ suitable to be used in a composition step the low-degree test from the previous chapter is crucial. We restate this result here so that the reader does not have to look back into Chapter 4. After the statement we illuminate its main features for the further ongoing.

Theorem (restatement of Theorem 4.2)**.** *Let $d \in \mathbb{N}$, $h = 10^{15}$, $k \geq \frac{3}{2}(2h+1)$ and let F be a finite field with $q := |F|$ being a prime number larger than $10^4(2hkd+1)^3$. There exists a probabilistic verification algorithm in the BSS-model of computation over the reals with the following properties:*

a) The verifier gets as input a function value table of a multivariate function $f : F^k \to \mathbb{R}$ *and a proof string* π *consisting of at most* $|F|^{2k}$ *segments (blocks). Each segment consists of* $2hkd+k+1$ *real components. The verifier generates* $O(k \cdot \log q)$ *random bits and inspects* $O(1)$ *many positions in the table for* f *as well as* $O(1)$ *many segments in the proof string* π *in order to make its decision. The running time of the verifier is polynomially bounded in the quantity* kd.

b) For every function value table representing a trigonometric max-degree d *polynomial there exists a proof string such that the verifier accepts with probability* 1.

c) For any $0 < \epsilon < 10^{-19}$ *and for every function value table whose distance to a closest max-degree* $\tilde{d} := 2hkd$ *polynomial is at least* 2ϵ *the probability that the verifier rejects is at least* ϵ*, no matter what proof string is given. Here, for two functions* $f, g : F^k \mapsto \mathbb{R}$ *their distance is defined as* $dist(f,g) := \frac{1}{|F^k|} \cdot |\{x \in F^k | f(x) \neq g(x)\}|$.

The theorem says that there exists a $(k \log |F|, kd))$-restricted verification procedure that receives as input a function value table $f : F^k \to \mathbb{R}$ and a proof π, accepts if f is a max-degree d polynomial and rejects with high probability if it is not close to a max-degree $2 \cdot 10^{15} kd$ polynomial. Note that in the statement above we do not only count the queries into π but also those into the table or f. The reason is that in the later use of the test both f and π are parts of the (expected) proof that the trigonometric polynomial f codes a zero of a given polynomial system. The crucial property of the procedure is that (f, π) are queried in a very structured form, i.e., only constantly many blocks of size at most $O(kd)$) are inspected. [1] Though the test presented in [36] achieves the same total bound on queries into a pair (f, π) verifying that π proves f to be close to an algebraic low-degree polynomial, the latter is not structured because it makes $O(kd)$ questions also into the table for f. The improved structure is essentially necessary for using the verifier in the composition part. Let us already mention here that below a potential zero $x \in \mathbb{R}^n$ of a QPS instance is coded via a trigonometric polynomial with the parameters k, d, q specified as $k := \lceil \frac{\log n}{\log \log n} \rceil, d := O(\log n)$ and $|F| := O(\log^6 n)$, respectively. Then the above verifier is $(\log(n), \text{polylog}(n))$-restricted.

5.3 The correctness test

Our goal in this section is to design a $(\log(n), \text{polylog}(n))$-restricted segmented verifier for $\text{NP}_{\mathbb{R}}$. It will then be an outer verifier in the verifier composition procedure (to be developed as well) to prove the $\text{PCP}_{\mathbb{R}}$ theorem. So far we have a segmented low-degree test for trigonometric polynomials at hand. In order to use it for the final verifier a sum-check procedure will be designed. As said already also this procedure must be in segmented form. In order to achieve this task we use the low-degree verifier for segmentation. This structure in principle is the same in the Arora et al. proof. However, new difficulties arise here due to our framework. A direct segmentation of arbitrary real verifiers using the

[1] in f each such block only has one real component because the test asks a constant number of function values only.

low-degree verifier fails because composing a trigonometric polynomial with an algebraic polynomial does not result in a low-degree polynomial whereas the composition of two algebraic polynomials does result in a low-degree polynomial. We overcome this restriction by first designing a segmented correctness test. Such a test is unnecessary in the Turing model because there the (algebraic) low-degree test has such a good structure already that a correctness test is not needed to prove segmentation. The correctness test allows to achieve a weak form of segmentation also for a sum-check procedure. This finally leads to a segmented $(\log(n), \text{polylog}(n))$-restricted verifier.

5.3.1 The test

The starting point for this subsection is the difference between error detection and error correction. Suppose throughout the following that f has passed the test of Theorem 4.2. Thus, f is close to a trigonometric polynomial $\tilde{f}$ of max-degree $\tilde{d} = 2 \cdot 10^{15}kd$; therefore, for a randomly chosen point x with high probability $f(x)$ and $\tilde{f}(x)$ coincide. However, if we fix an x_0 it still might be the case that $f(x_0)$ is false. Error correction now means to be able to correct the value with high probability; for example, the Walsh-Hadamard code used in the design of long transparent proofs allows such a correction.

For our purposes in the low-degree setting it will be sufficient to detect such an error with high probability for any fixed x_0. Thus we want to design a verifier V which works on a given *fixed* x_0, a table for f and an additional proof string and rejects with high probability if $f(x_0) \neq \tilde{f}(x_0)$.

An easy approach for designing such a V is the following. Assume that $f(x_0) \neq \tilde{f}(x_0)$.

The verifier chooses a random line ℓ passing through x_0 and expects the proof string to contain a univariate trigonometric polynomial p for ℓ. The verifier checks whether p and f have the same value in x_0. If not V rejects directly. If yes, then it means that p is different from the restriction of $\tilde{f}$ to ℓ and since they are both polynomials of low degree they actually disagree on most points on ℓ. With high probability this will be exposed by the following procedure. The verifier chooses a random point $x_{rand} \in \ell$ different from x_0 and checks whether p and f have the same value in x_{rand}. We already know that with high probability $\tilde{f}$ disagrees with p on x_{rand} and since x_{rand} is uniformly distributed over $F^k \setminus \{x_0\}$ and f is close to $\tilde{f}$ it follows that with high probability $f(x_{rand}) = \tilde{f}(x_{rand})$. Thus with high probability it holds that f and p disagree on x_{rand}, in which case the verifier rejects.

The problem with this approach is the following: Since above a random point was chosen the proof string needs to contain univariate restrictions with respect to all lines through F^k. But opposed to the case of algebraic polynomials in the trigonometric setting the degrees of such restrictions can get much larger than the degree of the corresponding multivariate polynomials because they might get an additional factor of size $\Omega(kq)$. So we need a set of lines through F^k that result in univariate restrictions of lower degree only. In fact, the resulting degrees will be bounded by $O(kd\sqrt{q})$ which turns out to suffice, see also Remark 5.4 below. A new problem occurs that way: How do we get a randomly distributed point if the set of used directions is restricted? This is solved as follows. V

sets up a random process by choosing constantly many times lines randomly from this set together with random points x_i on those lines. The above argument concerning x_0 and x_{rand} now is iteratively applied to the univariate polynomials along the lines through (x_i, x_{i+1}). The key point for V to work correctly is the observation that after finitely many steps (actually, 24 steps) the resulting point x_{24} in the process is almost randomly distributed in F^k. Most of the work below is devoted to demonstrate that this holds for the set of lines that we will construct.

Now towards the details. First, the correctness test shall be described. Let parameters $k, d, \tilde{d}, q$ be as in Theorem 4.2. The following objects are needed in the test procedure.

Definition 5.1. *a) For $F := \{0, 1, \ldots, q-1\}$ define the set $W \subset F$ as*

$$W := \{r \in F \mid |r| \leq \lfloor \sqrt{q} \rfloor\}.$$

Here, the absolute value of $r \in F$ is defined as $|r| := \begin{cases} r & \text{if } r < \frac{q}{2} \\ q - r & \text{if } r > \frac{q}{2} \end{cases}$.

b) For $x, v \in F^k$ denote by $\ell_{x,v} := \{x + tv | t \in F\}$ the line in F^k through x with direction v.

The set of lines that we will use is the set of lines $\ell_{x,v}$ with $x \in F^k$ and $v \in W^k$, so that if f is a max-degree d polynomial, then the restriction of f to any such line has degree at most $k\lfloor \sqrt{q} \rfloor d$ which is much smaller than q. Since $|W^k| = (2\lfloor \sqrt{q} \rfloor + 1)^k$ the set contains $q^{k-1} \cdot (2\lfloor \sqrt{q} \rfloor + 1)^k$ lines. Here we consider $\ell_{x,v}$ to be the same as $\ell_{x+v,v}$, but to be different from $\ell_{x,cv}$ for some $c \in F \setminus \{1\}$.

Correctness Test: Let $k, q, d, \tilde{d}, F, W$ be as above.

Input: A point $x_0 \in F^k$ and a function $f : F^k \to \mathbb{R}$, given by a table of its values. We suppose that f has passed without error the test behind Theorem 4.2 and denote by $\tilde{f}$ the unique polynomial of max-degree $\tilde{d} = 2 \cdot 10^{15} kd$ that is δ-close to f for small enough $0 < \delta$;

a proof string containing a list of univariate trigonometric polynomials $p_{x,v}$ of degree $k\lfloor \sqrt{q} \rfloor d$ defined for each line $\ell_{x,v}$ with direction in W^k and specified by its $2k\lfloor \sqrt{q} \rfloor d + 1$ coefficients. [2]

a) For all $0 \leq i \leq 23$ pick uniformly and independently a direction $v_i \in W^k$ and a random point $t_i \in F \setminus \{0\}$; set $x_{i+1} := x_i + t_i \cdot v_i$;

b) check whether $f(x_i) = p_{x_i,v_i}(0)$ and $f(x_{i+1}) = p_{x_i,v_i}(t_i)$. If for all i equality holds accept, otherwise reject.

[2] There is a certain ambiguity in representing such a polynomial $p_{x,v}$ because different points on the line can be used and different vectors from W^k might result in the same line. This is not a problem since one can efficiently switch between those representations. Below, when we evaluate such a polynomial in a point t^* we silently assume the parametrization induced by the x, v mentioned.

The same line might have several representations using elements from W^k. We are interested in the uniform distribution among all those representations; thus, the probability of picking one is proportional to the number of representations it has with respect to different elements in W^k.

The verifier performing the correctness test makes a random walk through F^k along lines with direction in W^k. Each x_i is a random point on a random line from W^k. Then it is checked whether the table for f corresponds to the value of the univariate polynomial p_{x_i,v_i} given as part of the input. In an ideal situation $f = \tilde{f}$ is a polynomial of max-degree d and the p_{x_i,v_i} are the univariate restrictions of f to the lines in W^k. As we shall see the final point x_{24} is almost uniformly distributed in F^k, no matter how x_0 looks like. Thus, finally the test nearly does the same as the straightforward approach described above which directly picks a random x_{rand}. However, this time the degrees of the corresponding univariate polynomials remain small.

Theorem 5.2 (Correctness Test)**.** *With the above notations and assumptions a verifier performing the Correctness Test satisfies the following:*

a) if $f \equiv \tilde{f}$ is a max-degree d polynomial and the $p_{x,v}$ represent the correct restrictions of f to line $\ell_{x,v}$, then the verifier accepts with probability 1;

b) if $f(x_0) \neq \tilde{f}(x_0)$, then the verifier accepts with probability at most $\delta + \frac{25}{\sqrt[4]{q}} + \frac{48\sqrt{q}k\tilde{d}}{q-1}$;

c) the verifier is $(k\log(q), k\lfloor\sqrt{q}\rfloor d)$-restricted; it inspects $O(1)$ many values of f and $O(1)$ many polynomials $p_{x,v}$, i.e., segmented blocks of length $2k\lfloor\sqrt{q}\rfloor d + 1 = O(kd)$. Its running time is polynomially bounded in $k\lfloor\sqrt{q}\rfloor d$.

The proof of b) splits in two parts, the second of which (Proposition 5.5) is the hard one and is treated in the next subsection. Let us first extract the more easy arguments involved.

Lemma 5.3. *Suppose under the assumption of Theorem 5.2 that $f(x_0) \neq \tilde{f}(x_0)$. Then the probability that the verifier accepts the correctness test is bounded from above by the sum $Pr(f(x_{24}) \neq \tilde{f}(x_{24})) + \frac{48\sqrt{q}k\tilde{d}}{q-1}$.*

Proof. For $0 \leq i \leq 23$ define events $A_i \equiv p_{x_i,v_i}(0) = f(x_i) \wedge p_{x_i,v_i}(t_i) = f(x_{i+1}), B_i \equiv f(x_i) \neq \tilde{f}(x_i) \wedge f(x_{i+1}) = \tilde{f}(x_{i+1})$ as well as $B_{24} \equiv f(x_{24}) \neq \tilde{f}(x_{24})$. Then by definition V accepts iff $\bigcap_{i=0}^{23} A_i$ holds. However, the latter is a subset of $B_{24} \cup \bigcup_{i=0}^{23}(A_i \cap B_i)$: by assumption $f(x_0) \neq \tilde{f}(x_0)$, thus either $A_0 \subseteq B_0$ (in which case the inclusion is proven) or $f(x_1) \neq \tilde{f}(x_1)$. Similarly, in that case either $A_1 \subseteq B_1$ (and we are done) or $f(x_2) \neq \tilde{f}(x_2)$. We continue with this argument until finally either $A_{23} \subseteq B_{23}$ or $f(x_{24}) \neq \tilde{f}(x_{24})$. But this is event B_{24}.

Therefore, it is $Pr(V \text{ accepts}) = Pr(\bigcap_{i=0}^{23} A_i) \leq Pr(B_{24}) + \sum_{i=0}^{23} Pr(A_i \cap B_i)$. The probabilities $Pr(A_i \cap B_i)$ are easily bounded: if p_{x_i,v_i} disagrees with $\tilde{f}$ in x_i, then as different

univariate polynomials of degree at most $2\sqrt{q}k\tilde{d}$ they agree in at most $2\sqrt{q}k\tilde{d}$ many values of F. But x_{i+1} is chosen uniformly from $F \setminus \{x_i\}$, so the probability that p_{x_i,v_i} and $\tilde{f}$ agree in x_{i+1} is at most $\frac{2\sqrt{q}k\tilde{d}}{q-1}$. The lemma follows. □

Remark 5.4. *The proof makes clear why univariate restrictions with not too high a degree are necessary, i.e., why all the efforts to show that W^k is an appropriate set of directions are needed.*

It thus remains to bound $Pr(f(x_{24}) \neq \tilde{f}(x_{24}))$. To do so it suffices to show that x_{24} will be almost uniformly distributed in F^k. In that case the above probability will be small because we assume that f and $\tilde{f}$ are δ-close.

Proposition 5.5. *With the notation of the correctness test it is*

$$\forall y \in F^k : \ Pr(\sum_{i=0}^{23} t_i v_i = y) \geq \frac{1}{q^k} \cdot (1 - \frac{25}{\sqrt[4]{q}}) \ .$$

Proof. Below after proof of Theorem 5.2. □

Proof. (of Theorem 5.2) Items a) and c) are obvious. For b) note that the uniform distribution on F^k is invariant under additive shifts, so Proposition 5.5 implies that for all $y \in F^k$ we also have $Pr(y = x_{24} := x_0 + \sum_{i=0}^{23} t_i v_i) \geq \frac{1}{q^k} \cdot (1 - \frac{25}{\sqrt[4]{q}})$. By assumption f and $\tilde{f}$ are δ-close, thus $Pr(f(x_{24}) \neq \tilde{f}(x_{24})) \leq 1 - (1-\delta)(1 - \frac{25}{\sqrt[4]{q}}) \leq \delta + \frac{25}{\sqrt[4]{q}}$. This and Lemma 5.3 imply the theorem. □

5.3.2 Proof of Proposition 5.5

The distributions of the coordinates of $y = \sum_{i=0}^{23} t_i v_i$ are independent of each other. Thus it suffices to show that a single coordinate y_ℓ is almost uniformly distributed in F. More precisely, we show that there exists a set $T \subseteq \{F^k \setminus \{0\}\}^{24}$ of size $|T| \geq (1 - \frac{24}{\sqrt[4]{q}})(q-1)^{24}$ such that for every $(t'_0, \ldots, t'_{23}) \in T$ and $s \in F$ we have $\Pr[y_\ell = s | t_0 = t'_0, \ldots, t_{23} = t'_{23}] \geq \frac{1}{q}(1 - \frac{1}{q^2})$. So basically we fix the t_i here and take the probability only over the v_i. Given our choices for k and q it follows having such a set T at hand that

$$\begin{aligned}
\Pr[y = \textstyle\sum_{i=0}^{23} v_i t_i] &\geq \Pr[(t_0, \ldots, t_{23}) \in T] \cdot \tfrac{1}{q^{24}}(1 - \tfrac{1}{q^2})^{24} \\
&\geq \left(1 - \tfrac{24}{\sqrt[4]{q}}\right) \cdot \tfrac{(q-1)^{24}}{(q-1)^k} \cdot \tfrac{1}{q^{24}}(1 - \tfrac{1}{q^2})^{24} \\
&\geq \left(1 - \tfrac{24}{\sqrt[4]{q}}\right) \cdot \tfrac{1}{q^k} \cdot (1 - \tfrac{1}{q^2})^{24} \text{ (because } k > 24) \\
&\geq \left(1 - \tfrac{24}{\sqrt[4]{q}}\right) \cdot (1 - \tfrac{24}{q^2}) \cdot \tfrac{1}{q^k} \text{ (Bernoulli)} \\
&\geq \tfrac{1}{q^k}(1 - \tfrac{25}{\sqrt[4]{q}}) \text{ (by our choice of } q \text{ it is } q^2 > 24\sqrt[4]{q})
\end{aligned}$$

which is what we need to show.

It remains to construct a set T with the properties mentioned above. Towards this end consider a fixed $t' = (t'_0, \ldots, t'_{23}) \in (F \setminus \{0\})^{24}$ and define random variables $Q_i, i = 0, \ldots, 24$ yielding the first coordinate of $\sum_{j=0}^{i-1} t'_j v_j$, where $Q_0 := 0$. Thus only the choices of the directions v_j remain random. Obviously, for every $s \in F, i = 1, \ldots, 24$ it holds

$$\Pr(Q_i = s) \quad = \quad \frac{1}{|W|} \sum_{s' \in W} \Pr(Q_{i-1} = s - s' \cdot t'_{i-1}) \tag{5.1}$$

The main idea for showing the existence of the set T is to bound for all $s \in F$ the difference between the probability that $Q_{24} = s$ and the uniform probability of s which is $\frac{1}{q}$. We do so by proving that the entire sum $\sum_{s \in F} \left(Pr(Q_{24} = s) - \frac{1}{q} \right)^2$ of the squares of those distances remains small, so all of its addends are small as well. This is achieved by a technical result given as Claim 1 below. In a sense for any function $g : F \mapsto \mathbb{R}$ it deals with the single steps of the random walk used in the correctness test, see the left hand side of the inequality below. We first state the claim, prove how it implies the existence of T and finally prove the claim.

Claim 1 : Let $g : F \mapsto \mathbb{R}$ be an arbitrary function and let $\mu := \frac{1}{q} \sum_{s \in F} g(s)$. Then for at least a $(1 - \frac{1}{\sqrt{|W|}})$-fraction of $t \in F \setminus \{0\}$ it holds that

$$\sum_{s \in F} \left(\frac{1}{|W|} \sum_{s' \in W} g(s - s' \cdot t) - \mu \right)^2 \leq \frac{1}{\sqrt{|W|}} \sum_{s \in F} (g(s) - \mu)^2 .$$

Assuming Claim 1 to be true the existence of a suitable set T can be shown as follows. For a fixed $t' = (t'_0, \ldots, t'_{23})$ apply equation (1) and the claim iteratively to the functions $g_i(s) := Pr(Q_i = s)$ and $\mu_i := \frac{1}{q} \sum_{s \in F} g_i(s) = \frac{1}{q}$ for all $0 \leq i \leq 23$. If all t'_i belong to the corresponding fraction of values for g_i such that Claim 1 is satisfied, then this yields the inequality chain

$$\begin{aligned}
\sum_{s \in F} \left(Pr(Q_{24} = s) - \tfrac{1}{q} \right)^2 \quad & \underbrace{=}_{(1)} \quad \sum_{s \in F} \left(\tfrac{1}{|W|} \sum_{s' \in W} \Pr(Q_{23} = s - s' \cdot t'_{23}) - \tfrac{1}{q} \right)^2 \\
& \underbrace{\leq}_{Claim1} \quad \tfrac{1}{\sqrt{|W|}} \sum_{s \in F} \left(Pr(Q_{23} = s) - \tfrac{1}{q} \right)^2 \\
& \underbrace{=}_{(1)} \quad \tfrac{1}{\sqrt{|W|}} \sum_{s \in F} \left(\tfrac{1}{|W|} \sum_{s' \in W} \Pr(Q_{22} = s - s' \cdot t'_{22}) - \tfrac{1}{q} \right)^2 \\
& \underbrace{\leq}_{Claim1} \quad \tfrac{1}{\sqrt{|W|}^2} \sum_{s \in F} \left(Pr(Q_{22} = s) - \tfrac{1}{q} \right)^2 \\
& \vdots \\
& \leq \quad \tfrac{1}{|W|^{12}} \sum_{s \in F} \left(Pr(Q_0 = s) - \tfrac{1}{q} \right)^2 \; = \; \tfrac{1}{|W|^{12}} \cdot ((1 - \tfrac{1}{q})^2 + \tfrac{q-1}{q^2}) \leq (4q)^{-6},
\end{aligned}$$

where the final line follows because $Q_0 = 0$ and $|W| = 2\sqrt{q}+1$. Define T as the set of all $(t'_0, \ldots, t'_{23}) \in (F \setminus \{0\})^{24}$ where t'_i satisfies Claim 1 for g_i. Then Claim 1 implies on one hand side the cardinality bound $|T| \geq (1 - \frac{1}{\sqrt{|W|}})^{24}(q-1)^{24} \geq (1 - \frac{24}{\sqrt[4]{q}})(q-1)^{24}$. On the other hand for each $t' \in T$ using t' in the random walk the above calculations show for each $s \in F$ that $\left(\Pr(Q_{24} = s) - \frac{1}{q}\right)^2 \leq (4q)^{-6}$ and therefore $\Pr(Q_{24} = s) \geq \frac{1}{q}(1 - \frac{1}{q^2})$. Thus, both required properties for T hold and the proposition follows.

It remains to prove Claim 1. Here, a second technical result is needed:

Claim 2: For elements $u', u'' \in F$ define

$$M(u', u'') := \left| \left\{ (t, s) \in (F \setminus \{0\}) \times F \mid \frac{s - u'}{t} \in W \text{ and } \frac{s - u''}{t} \in W \right\} \right|.$$

Then it is

$$M(u', u'') = \begin{cases} (|W| - 1) \cdot |W| & \text{if } u' \neq u'' \\ (q-1) \cdot |W| & \text{if } u' = u'' \end{cases}.$$

Thus $M(u', u'')$ is the number of pairs (t, s) for which there exist (unique) $s', s'' \in W$ such that $s - ts' = u'$ and $s - ts'' = u''$. This is needed below.

Proof of Claim 2: It is easy to see that $M(u', u') = (q-1) \cdot |W|$: in this case, for each choice of $t \in F \setminus \{0\}$ there are exactly $|W|$ suitable choices for s. Suppose then $u' \neq u''$. Split the set of suitable (t, s) according to the absolute value $|\frac{u'-u''}{t}|$, see Definition 5.1. Since for all $s', s'' \in W$ it holds that $|s' - s''| \leq |W| - 1$ it is clear that $\frac{s-u'}{t} \in W$ and $\frac{s-u''}{t} \in W$ yields $|\frac{u'-u''}{t}| \leq |W| - 1$. Therefore, if $u' \neq u''$ one obtains

$$M(u', u'') = \sum_{i=1}^{|W|-1} \left| \left\{ (t, s) \in (F \setminus \{0\}) \times F \mid \frac{s - u'}{t} \in W,\ \frac{s - u''}{t} \in W \text{ and } |\frac{u' - u''}{t}| = i \right\} \right|.$$

Setting $\tilde{t} := \frac{1}{t}$ and $\tilde{s} := \frac{s}{t}$ this simplifies to

$$M(u', u'') = \sum_{i=1}^{|W|-1} \left| \left\{ (\tilde{t}, \tilde{s}) \in (F \setminus \{0\}) \times F \mid \tilde{s} - \tilde{t}u' \in W,\ \tilde{s} - \tilde{t}u'' \in W \text{ and } |\tilde{t}u' - \tilde{t}u''| = i \right\} \right|.$$

Since F is a field for each $i = 1, \ldots, |W| - 1$ the condition $|\tilde{t}u' - \tilde{t}u''| = i$ is satisfied by exactly two values of $\tilde{t}$, namely $\tilde{t} = \frac{i}{u'-u''}$ and $\tilde{t} = \frac{q-i}{u'-u''}$. Now if $|\tilde{t}u' - \tilde{t}u''| = i$ it follows that for such an i and $\tilde{t}$ there are exactly $|W| - i$ values for $\tilde{s}$ satisfying $\tilde{s} - \tilde{t}u' \in W$ and $\tilde{s} - \tilde{t}u'' \in W$; this is true because W is an interval. Hence, for $u' \neq u''$ we have

$$M(u', u'') = \sum_{i=1}^{|W|-1} 2 \cdot (|W| - i) = (|W| - 1) \cdot |W|$$

concluding the proof of Claim 2.

We are finally ready to give the
Proof of Claim 1: Let $g : F \to \mathbb{R}$ and $\mu := \frac{1}{q} \sum_{s \in F} g(s)$.

The idea is to compute the sum of $\sum_{s \in F} \left(\frac{1}{|W|} \sum_{s' \in W} g(s - s't) - \mu \right)^2$ over all $t \in F \setminus \{0\}$ and show that it is very small. From that we conclude that for most t the value $\sum_{s \in F} \left(\frac{1}{|W|} \sum_{s' \in W} g(s - s't) - \mu \right)^2$ is smaller than $\frac{1}{\sqrt{|W|}} \sum_{s \in F} (g(s) - \mu)^2$. It is

$$
\begin{aligned}
& \sum_{t \in F \setminus \{0\}} \sum_{s \in F} \left(\frac{1}{|W|} \sum_{s' \in W} g(s - s't) - \mu \right)^2 \\
= \; & \frac{1}{|W|^2} \sum_{t \in F \setminus \{0\}} \sum_{s \in F} \left(\sum_{s' \in W} (g(s - ts') - \mu) \right)^2 \\
= \; & \frac{1}{|W|^2} \sum_{t \in F \setminus \{0\}} \sum_{s \in F} \sum_{s', s'' \in W} (g(s - ts') - \mu) \cdot (g(s - ts'') - \mu) \\
= \; & \frac{1}{|W|^2} \sum_{u', u'' \in F} (g(u') - \mu) \cdot (g(u'') - \mu) \cdot M(u', u'') \\
= \; & \frac{|W| - 1}{|W|} \sum_{u', u'' \in F, u' \neq u''} (g(u') - \mu) \cdot (g(u'') - \mu) + \frac{q - 1}{|W|} \sum_{u' \in F} (g(u') - \mu)^2 \\
= \; & \frac{|W| - 1}{|W|} \underbrace{\sum_{u', u'' \in F} (g(u') - \mu) \cdot (g(u'') - \mu)}_{= 0 \text{ because } \mu = \frac{1}{q} \sum_{s \in F} g(s)} + \frac{q - |W|}{|W|} \sum_{u' \in F} (g(u') - \mu)^2 \\
= \; & \frac{q - |W|}{|W|} \sum_{u' \in F} (g(u') - \mu)^2.
\end{aligned}
$$

It follows immediately that there can be at most a $(\frac{1}{\sqrt{|W|}})$-fraction of values $t \in F \setminus \{0\}$ for which $\sum_{s \in F} \left(\frac{1}{|W|} \sum_{s' \in W} g(s - s't) - \mu \right)^2 \geq \frac{1}{\sqrt{|W|}} \sum_{s \in F} (g(s) - \mu)^2$. Claim 1 and Proposition 5.5 follow. □

5.4 Segmented almost transparent proofs for $\mathrm{NP}_{\mathbb{R}}$

In the Arora et al. proof the next important step in designing a $(\log(n), \mathrm{polylog}(n))$-restricted verifier for NP by using a low-degree test is a so-called sum-checking procedure. Very roughly, such a procedure probabilistically checks whether an expression of the form $\sum_{z_1, \ldots, z_k = 0}^{h} \tilde{g}(z)$ for some specified $h \in F$ vanishes. Below we apply all the above to an

instance of size n for an $\mathrm{NP}_{\mathbb{R}}$-complete problem and get a similar problem where $\tilde{g}$ is a trigonometric polynomial of (low) degree $O(\log n)$ and $h = \lfloor \log n \rfloor, k = \lceil \frac{\log n}{\log \log n} \rceil$. Then the sum has $h^k = \Omega(n)$ many terms, so computing it exactly would require to query $\tilde{g}$ in too many arguments. Sum-checking does the evaluation randomly in order to reduce the query complexity to polylog(n). In addition, this randomized evaluation has to be segmented for using it in the rest of the proof. This creates serious problems in our approach which we deal with next.

5.4.1 A technical condition: Summation property

In order to do a sum-check the idea is as follows. We do not work directly with $\tilde{g}$, but with a polynomial $\tilde{f} : F^{2k} \mapsto \mathbb{R}$ which is related to $\tilde{g}$ in a very special manner through a summation property. This technical property allows to express the sum-checking problem for $\tilde{g}$ as evaluation of $\tilde{f}$ in a single point. The disadvantage is that it must be checked as well whether $\tilde{f}$ has the summation property. Of course, this verification needs to be done in segmented form as well.

We continue as follows: First, the summation property relating two polynomials $\tilde{g}$ and $\tilde{f}$ is defined; if satisfied it directly transfers sum-checking for $\tilde{g}$ to a single-point evaluation of $\tilde{f}$. Next, we prove why for polynomials $\tilde{g}$ such an $\tilde{f}$ always exists. Finally, it is proved that the summation property for $\tilde{f}$ can be tested within the allowed resources.

Definition 5.6. *a) For a k-tuple $z = (z_1, \ldots, z_k)$, a component $1 \leq j \leq k$ and a $t \in F$ let $P_t^j(z) := (z_1, \ldots, z_{j-1}, t, z_{j+1}, \ldots z_k)$, i.e., the j-th component is assigned the value t.*

b) Let $\tilde{g} : F^k \to \mathbb{R}$ be a max-degree $\tilde{d}$ polynomial and let $h \in F$. We say that a max-degree $\tilde{d}$ polynomial $\tilde{f} : F^{2k} \to \mathbb{R}$ satisfies the summation property with respect to $\tilde{g}$ and h *if the following conditions hold:*

a) for every $x, y \in F^k, j \in \{1, \ldots, k\}$ it is $\frac{1}{h+1} \sum_{t=0}^{h} \tilde{f}(P_t^j(x), y) = \tilde{f}(x, P_1^j(y))$;

b) the restriction of $\tilde{f}$ to $F^k \times \{0\}^k$ equals $\tilde{g}$.

The summation property can be used to replace the sum-check for $\tilde{g}$ by an equality test for a single argument:

Lemma 5.7. *Let $h, \tilde{f}, \tilde{g}$ be as above. If the summation property holds for $\tilde{f}$ with respect to $\tilde{g}$ and h, then*

$$\frac{1}{(h+1)^k} \sum_{z_1, \ldots, z_k = 0}^{h} \tilde{g}(z) = \tilde{f}(0^k, 1^k) = \tilde{f}(x, 1^k) \text{ for every } x \in F^k.$$

Proof. Suppose conditions a) and b) from the definition to hold, then an easy calculation shows

$$\begin{aligned}
\tilde{f}(0^k,1^k) &= \tilde{f}(0^k,P_1^kP_0^k1^k) = \frac{1}{h+1}\sum_{z_k=0}^{h}\tilde{f}(P_{z_k}^k0^k,P_0^k1^k)\\
&= \frac{1}{h+1}\sum_{z_k=0}^{h}\tilde{f}(P_{z_k}^k0^k,P_1^{k-1}P_0^{k-1}P_0^k1^k)\\
&= \frac{1}{(h+1)^2}\sum_{z_{k-1}=0}^{h}\sum_{z_k=0}^{h}\tilde{f}(P_{z_{k-1}}^{k-1}P_{z_k}^k0^k,P_0^{k-1}P_0^k1^k)\\
&\vdots\\
&= \frac{1}{(h+1)^k}\sum_{z_1=0}^{h}\cdots\sum_{z_k=0}^{h}\tilde{f}(P_{z_1}^1\dots P_{z_k}^k0^k,P_0^1\dots P_0^k1^k)\\
&= \frac{1}{(h+1)^k}\sum_{z_1=0}^{h}\cdots\sum_{z_k=0}^{h}\tilde{f}(z_1,\dots,z_k,0^k)\\
&= \frac{1}{(h+1)^k}\sum_{z_1=0}^{h}\cdots\sum_{z_k=0}^{h}\tilde{g}(z_1,\dots,z_k).
\end{aligned}$$

□

Thus it is sufficient to develop a test for verifying condition a) of the summation property and do a Schwartz-Zippel equality test for condition b). We shall first show why all this is useful under certain assumptions. In the next subsection we then show why those assumptions can be satisfied.

The scenario needed further on and explained in Subsection 5.4.3 is as follows: We are given a black-box for computing a function $g : F^k \mapsto \mathbb{R}$ which is close to a max-degree $\tilde{d}$ trigonometric polynomial $\tilde{g} : F^k \mapsto \mathbb{R}$. Given suitable $h \in F$ an equation $\sum_{z_1=0}^{h}\dots\sum_{z_k=0}^{h}\tilde{g}(z_1,\dots,z_k) = 0$ should be verified. We shall specify later on how g and $\tilde{g}$ arise in the general framework of this chapter.

The following easy theorem shows how finally a sum-check can be done if an $\tilde{f}$ is available that satisfies the summation property with respect to $\tilde{g}$ and h. The difficult part will be to prove that the assumptions of the theorem can be satisfied.

Theorem 5.8. *Let $q,k,h,d,\tilde{d},g,\tilde{g}$ be as above. Suppose there exists a max-degree d polynomial $\tilde{f} : F^{2k} \mapsto \mathbb{R}$ such that $\tilde{f}$ satisfies the summation property with respect to $\tilde{g}$ and h. Suppose furthermore that there exists a segmented test for verifying condition a) of the summation property for $\tilde{f}$. Then there is a segmented verification procedure for the sum check $\sum_{z_1=0}^{h}\dots\sum_{z_k=0}^{h}\tilde{g}(z_1,\dots,z_k) = 0$. The resources the latter verifier needs are those for the*

test verifying condition a) plus $O(k \log q)$ random bits plus $O(k\lfloor\sqrt{q}\rfloor d)$ proof components.

Proof. Beside using the test for checking condition a) of the summation property the verifier expects the proof to contain a function value table for an $f : F^{2k} \to \mathbb{R}$. It performs the low-degree test from Theorem 4.2 on f to verify that it is close to a max-degree $\tilde{d} = 2 \cdot 10^{15}kd$ polynomial $\tilde{f}$. Then, it chooses a random $x \in F^k$ and performs the correctness test from Theorem 5.2 on $f(x, 0^k)$ and on $g(x)$. Since $x \mapsto \tilde{f}(x, 0^k)$ and $x \mapsto \tilde{g}(x)$ are polynomials of low degree, it follows from the usual Schwartz-Zippel technique that if they are not identical, then with high probability $\tilde{f}(x, 0^k) \neq \tilde{g}(x)$. It also holds with high probability that $g(x) = \tilde{g}(x)$ and if both are the case, then either $g(x) \neq f(x, 0^k)$ or the correctness test rejects with high probability. Hence, if no rejection occurs, then we may assume that for all $x \in F^k$, $\tilde{f}(x, 0^k) = \tilde{g}(x)$.

A sum-check to verify whether $\sum_{z_1=0}^{h} \cdots \sum_{z_k=0}^{h} \tilde{g}(z_1, \ldots, z_k) = s$ for $s \in \mathbb{R}$ now can be done by simply evaluating $(h+1)^k \cdot \tilde{f}(0^k, 1^k)$ using the correctness test and comparing the result with s. The resource bounds follow from those for the low-degree test, the correctness test, and the Schwartz-Zippel test. □

So next it has to be shown that the assumptions of the theorem can be satisfied.

5.4.2 Existence of $\tilde{f}$; testing summation property

The next technical lemma shows that for an arbitrary polynomial $\tilde{g}$ and $h \in F$ such an $\tilde{f}$ exists.

Lemma 5.9. *Let $\tilde{g} : F^k \to \mathbb{R}$ be a max-degree d polynomial and let $h \in F$. There exists a max-degree d polynomial $\tilde{f} : F^{2k} \to \mathbb{R}$ satisfying the summation property with respect to $\tilde{g}$ and h.*

Proof. Given $\tilde{g} : F^k \to \mathbb{R}$ of degree d and $h \in F$ we define $\tilde{f}$ in several steps. First, requirement b) of the summation property forces us to put $\tilde{f}(x, 0) := \tilde{g}(x)$ for all $x \in F^k$. As next step we extend the definition of $\tilde{f}$ to the domain $F^k \times \{0,1\}^k$ as follows. For $x \in F^k$ and $y \in \{0,1\}^k$ define $\tilde{f}(x, y)$ to be the average of $\tilde{g}$ over those coordinates j where $y_j = 1$. For example, if $y_2 = y_3 = y_5 = 1$ and all other coordinates of y are 0, then $\tilde{f}(x, y) := \frac{1}{(h+1)^3} \sum_{z_2,z_3,z_5=0}^{h} \tilde{g}(x_1, z_2, z_3, x_4, z_5, x_6, \ldots, x_k)$. More formally, this yields

$$\tilde{f}(x, y) := \frac{1}{(h+1)^k} \sum_{z \in \{0,\ldots,h\}^k} \tilde{g}(x \cdot \bar{y} + z \cdot y) \quad \forall x \in F^k, y \in \{0,1\}^k,$$

where $\bar{y} := (1 - y_1, \ldots, 1 - y_k)$ and $x \cdot y := (x_1 y_1, \ldots, x_k y_k)$. The definition gives (again) $\tilde{f}(x, 0) = \tilde{g}(x)$; moreover, for all $x \in F^k$ $\tilde{f}(0^k, 1^k) = \tilde{f}(x, 1^k) = \frac{1}{(h+1)^k} \sum_{z_1,\ldots,z_k=0}^{h} \tilde{g}(z)$ from which condition c) of the summation property follows.

Now extend $\tilde{f}$ to the full domain F^{2k}. Consider a polynomial $\tau : F^k \times \{0,1\}^k \to \mathbb{R}$ with the property that for $y, u \in \{0,1\}^k$ it is $\tau(y,u) := \begin{cases} 1 & \text{if } y = u \\ 0 & \text{if } y \neq u \end{cases}$. Such a τ can be defined as

$$\tau(y,u) := \left(\prod_{i:u_i=0} (1 - \widetilde{\sin}(y_i)) \right) \left(\prod_{i:u_i=1} \widetilde{sin}(y_i) \right)$$

where $\widetilde{\sin}(y_i) := \frac{\sin(2\pi y_i/q)}{\sin(2\pi/q)}$; then for $y_i \in \{0,1\}$ it is $\widetilde{\sin}(y_i) = y_i$.

Now for all $x, y \in F^k$ put

$$\tilde{f}(x,y) := \sum_{u \in \{0,1\}^k} \left[\frac{\tau(y,u)}{(h+1)^k} \sum_{z \in \{0,\ldots,h\}^k} \tilde{g}(x \cdot \bar{u} + z \cdot u) \right].$$

Due to the properties of τ this definition agrees with the definition of $\tilde{f}$ on $F^k \times \{0,1\}^k$. Furthermore, since $\tilde{g}$ has degree d in x and τ has degree 1 in y the degree of $\tilde{f}$ is d.

It remains to show that $\tilde{f}$ satisfies the summation condition for $h \in F$. For notational simplicity for $x \in F^k$, $u \in \{0,1\}^k$ define

$$\gamma(x,u) := \frac{1}{(h+1)^k} \sum_{z \in \{0,\ldots,h\}^k} \tilde{g}(x \cdot \bar{u} + z \cdot u),$$

so that $\tilde{f}(x,y) = \sum_{u \in \{0,1\}^k} \tau(y,u) \cdot \gamma(x,u)$. The following equations for all $j \leq k$, $x, y \in F^k$ and $u \in \{0,1\}^k$ are useful in the subsequent calculations:

$$\frac{1}{h+1} \sum_{t=0}^{h} \gamma(P_t^j x, u) = \gamma(x, P_1^j u) \tag{5.2}$$

$$\tau(y, P_0^j u) + \tau(y, P_1^j u) = \tau(P_1^j y, P_1^j u) \tag{5.3}$$

$$\tau(P_1^j y, P_0^j u) = 0 \tag{5.4}$$

The first of these equations can be derived by evaluating both sides once for $u_j = 0$ and for $u_j = 1$. The other two immediately follow from the definitions. Now for all $1 \leq j \leq k$

$$\begin{aligned}
\tfrac{1}{h+1} \sum_{t=0}^{h} \tilde{f}(P_t^j x, y) &= \tfrac{1}{h+1} \sum_{t=0}^{h} \sum_{u \in \{0,1\}^k} \tau(y,u) \cdot \gamma(P_t^j x, u) \\
&= \sum_{u \in \{0,1\}^k} \tau(y,u) \cdot \gamma(x, P_1^j u) && \text{by (5.2)} \\
&= \sum_{u \in \{0,1\}^k : u_j = 1} (\tau(y, P_0^j u) + \tau(y, P_1^j u)) \cdot \gamma(x, P_1^j u) \\
&= \sum_{u \in \{0,1\}^k : u_j = 1} \tau(P_1^j y, P_1^j u) \cdot \gamma(x, P_1^j u) && \text{by (5.3)} \\
&= \sum_{u \in \{0,1\}^k} \tau(P_1^j y, u) \cdot \gamma(x,u) && \text{by (5.4) addend for } u_j = 0 \text{ is 0} \\
&= \tilde{f}(x, P_1^j y)
\end{aligned}$$

□

Testing validity of the summation property.

Our test for checking whether a given function $\tilde{f}$ satisfies condition a) from Definition 5.6 is based on the next lemma.

Lemma 5.10. *Suppose $|F| =: q \in \mathbb{N}, \tilde{d}, h < q$ and let $f : F^{2k} \to \mathbb{R}$ be 0.1-close to a max-degree $\tilde{d}$ polynomial $\tilde{f} : F^{2k} \to \mathbb{R}$. For every $i \in \{1, \ldots, 2k\}$ let $\tilde{f}_i : F^{2k} :\to \mathbb{R}$ be a degree d polynomial in its i-th variable; this means that $\tilde{f}_i$ restricted to a line in the i-th paraxial direction e_i is a polynomial of degree d. If for every $j \leq k$ the following three conditions hold:*

$$\Pr_{x,y \in F^k}\left(\frac{1}{h+1}\sum_{t=0}^{h} \tilde{f}_j(P_t^j x, y) = \tilde{f}_{k+j}(x, P_1^j y)\right) \geq 0.9 \tag{5.5}$$

$$\Pr_{x,y \in F^k}\Big(f(x,y) = \tilde{f}_j(x,y)\Big) \geq 0.9 \tag{5.6}$$

$$\Pr_{x,y \in F^k}\Big(f(x,y) = \tilde{f}_{k+j}(x,y)\Big) \geq 0.9, \tag{5.7}$$

then $\tilde{f}$ satisfies condition a) of the summation property for h.

Proof. From the assumptions, for $1 \leq i \leq 2k$ it follows $\Pr_{x,y \in F^k}\Big(\tilde{f}(x,y) = \tilde{f}_i(x,y)\Big) \geq 0.8$. Restricted to a line in the i-th paraxial direction e_i (e_i denoting the corresponding unit vector) $\tilde{f}$ is a degree $\tilde{d}$ polynomial and $\tilde{f}_i$ is a degree d polynomial. Hence, on such a line they either agree completely or they agree in at most $2\tilde{d}$ points. Assuming $q \geq 10\tilde{d}$ an easy calculation shows that $\tilde{f}$ and $\tilde{f}_i$ then disagree on at most a $\frac{1}{4}$-fraction of paraxial lines having the direction e_i (there are q^{k-1} such lines). Thus, for every $j \leq k$ there is a $\frac{3}{4}$-fraction of points $(x,y) \in F^{2k}$ such that $\tilde{f}$ and $\tilde{f}_j$ agree on the paraxial line in direction e_j through (x,y); the same holds for $\tilde{f}$ and $\tilde{f}_{k+j}$ with respect to lines into direction e_{k+j}. And for at least a $\frac{1}{2}$-fraction of points $(x,y) \in F^{2k}$ $\tilde{f}$ agrees with $\tilde{f}_j$ on the line in direction e_j through (x,y) and at the same time with $\tilde{f}_{k+j}$ on the line in direction e_{k+j} through (x,y). Using (5.5) it follows that

$$\Pr_{x,y \in F^k}\left(\frac{1}{h+1}\sum_{t=0}^{h} \tilde{f}(P_t^j x, y) = \tilde{f}(x, P_1^j y)\right) \geq 0.4. \tag{5.8}$$

Since $\tilde{f}$ is a max-degree $\tilde{d}$ polynomial, so are $(x,y) \mapsto \frac{1}{h+1}\sum_{t=0}^{h} \tilde{f}(P_t^j x, y)$ and $(x,y) \mapsto \tilde{f}(x, P_1^j y)$. By Schwartz-Zippel they either agree in at most a $\frac{2kd}{q}$ fraction of arguments or they are identical. Hence, by equation (5.8) and the choice of d, q they are identical. □

The lemma implies a test whether a given function $f : F^{2k} \to \mathbb{R}$ is close to a max-degree $\tilde{d}$ polynomial $\tilde{f}$ which satisfies condition a) of the summation property: use the low-degree test to verify closeness, then expect the proof in addition to contain the univariate

polynomials $\tilde{f}_i$ and check for all $1 \leq j \leq k$ whether (5.5), (5.6) and (5.7) hold. The problem is that this is not a segmented procedure because these checks have to be made for every j. Therefore we will now turn this test into a segmented one.

The idea is to encode all $\tilde{f}_i$ together into a single low-degree polynomial $s : F_{q'}^{k'} \to \mathbb{R}$. Here $F_{q'}$ denotes another finite field of suitable cardinality $q' :\geq 10^4(2 \cdot 10^{15}(2k+2)(q-1)(2k-1)+1)^3$. This cardinality allows us to apply Theorem 4.2 in the next lemma. We will view this as encoding a set of univariate degree d polynomials indexed by the paraxial lines in the (old) F^{2k}.This encoding is done as follows. Let $k' := 2k+2$. The coefficients of the degree d polynomial for the line into the i-th paraxial direction e_i through a point $x \in F^{2k}$ are encoded as the values of s on the $2d+1$ arguments

$$(i, m, x_1(i-1), x_2(i-2)+q-1, \ldots, x_{2k}(i-2k)+(2k-1)(q-1)).$$

Above the two first components are i and $m \in \{0, \ldots, 2d\}$, and the components belong to $\{0, \ldots, (q-1)(2k-1)\}$. Hence, for every such set of univariate polynomials mentioned above there exists a trigonometric polynomial $s : F_{q'}^{k'} \to \mathbb{R}$ of max-degree $(q-1)(2k-1)$ which corresponds to this set according to the encoding mentioned above. Note that although the pair (i, x) is not a unique specification for a line, all (i, x) corresponding to the same line give the same arguments on which s specifies the values of the corresponding degree d polynomial. This is due to the fact that points on a line into direction e_i only differ in their i-th component and this component in the above encoding is set to $x_i(i-i)+(i-1)(q-1) = (i-1)(q-1)$, i.e., it is independent of x_i.

Note as well that the encoding of all polynomials corresponding to lines through a fixed point $x \in F^{2k}$ lie in a single plane with the two directional vectors $(1, 0, x_1, \ldots, x_{2k})^T$ (for scalar i) and $(0, 1, 0, \ldots, 0)$ (for scalar m), respectively. This is used below to bound the degree of univariate restrictions.

Lemma 5.11. *There exists a segmented procedure to test whether a given function $f : F^{2k} \to \mathbb{R}$ is close to a low-degree polynomial $\tilde{f}$ satisfying the condition a) of summation property with respect to h.*

Proof. The verification procedure performs several independent tests. All of them are segmented according to previous results. First use Theorem 4.2 to test whether a given f is close to a max-degree $\tilde{d} := 2 \cdot 10^{15}kd$ polynomial $\tilde{f}$. Next, the proof certificate is expected to specify a function $s : F_{q'}^{k'} \to \mathbb{R}$ and the theorem is used once again to test that s is close to a max-degree $d' := 2 \cdot 10^{15}k'(q-1)(2k-1)$ polynomial $\tilde{s}$. Note that above we expect s to have max-degree $(q-1)(2k-1)$, so Theorem 4.2 results in max-degree d' for $\tilde{s}$.

Next, for every $x \in F^{2k}$ the proof certificate is required to specify a bivariate polynomial. In the ideal case it agrees with $s = \tilde{s}$ on the plane that contains the encodings of the polynomials corresponding to the paraxial lines through x.

Condition a) of the summation property of $\tilde{f}$ is now verified by repeating the following procedure a constant number of times: Select a random $x \in F^{2k}$ and query the bivariate polynomial describing the plane in $F_{q'}^{k'}$ that corresponds to x as described above. This

polynomial has at most max-degree $(1+2k(q-1))\cdot(q-1)(2k-1)$ because the directional vectors of the corresponding plane are $(1,0,x_1,\ldots,x_{2k})^T$ and $(0,1,0,\ldots,0)$. Thus it can be specified with a segment of length $[2(1+2k(q-1))\cdot(q-1)(2k-1)+1]^2$. First test whether this bivariate polynomial agrees with $\tilde{s}$ on the plane. By Schwartz-Zippel they either agree or they disagree in almost all arguments. So it suffices to test agreement for a constant number of arguments. Within this part of the procedure, the value of $\tilde{s}$ can be obtained using the correctness test and Theorem 5.2. Finally, the bivariate polynomial is used to obtain the univariate polynomials corresponding to the paraxial lines through x. These polynomials are used to test (5.5), (5.6) and (5.7) as described above before the theorem, but this time all performed tests are segmented. □

5.4.3 Almost transparent segmented short proofs for $\mathrm{NP}_{\mathbb{R}}$

Putting all the above together the following theorem can be proved. The setup for the sum-check is the same as for example in [36], so below we describe it briefly only in the proof.

Theorem 5.12. *For every problem $L \in \mathrm{NP}_{\mathbb{R}}$ there is a $(\log(n), \mathrm{polylog}(n))$-restricted real verifier accepting L. The verifier is segmented, i.e., it reads $O(1)$ many blocks of length $\mathrm{polylog}(n)$ from the proof certificate.*

Proof. Consider the $\mathrm{NP}_{\mathbb{R}}$-complete QPS problem. Suppose $\{p_1,\ldots,p_m\}$ to be a quadratic polynomial system with n variables, where every p_i depends on at most three variables. Let $h := \lfloor \log n \rfloor, d := \lfloor \log n \rfloor$, and $k = \lceil \frac{\log n}{\log\log n} \rceil$. Thus $(h+1)^k \geq n$ and the indices of the n variables can be addressed by elements from $\{0,1,\ldots,h\}^k$.

The verifier first expects the prover to present a potential zero of the system coded via a trigonometric polynomial $a : F^k \to \mathbb{R}$, where F is a finite field with $q := |F|$ a prime larger than $10^{50}k^3d^3$. A low-degree test (Theorem 4.2) with degree d is performed on the function value table for a. If the test accepts, then a is close to a max-degree $2\cdot 10^{15}kd$ polynomial $\tilde{a}$.

Next, the verifier tests whether the assignment for variables $x_1,\ldots,x_n$ coded by $\tilde{a}$ using the above addressing of variable indexes really gives a zero of the initial system.

Towards this aim it deterministically computes for all $i = 1,2,\ldots,m$ a max-degree d polynomial $\chi_i : F^{3k} \to \mathbb{R}$ such that if $x_1,x_2,x_3 \in \{0,1,\ldots,h\}^k$ correspond to the indices of those variables on which polynomial p_i depends, then $\chi_i(x_1,x_2,x_3) = 1$ and for all other elements $x \in \{0,1,\ldots,h\}^{3k}$ it is $\chi_i(x) = 0$.

Now the assignment coded by trigonometric polynomial a is a zero of p_i iff

$$\sum_{(x_1,x_2,x_3)\in\{0,1,\ldots,h\}^{3k}} \chi_i(x_1,x_2,x_3)\cdot(p_i(\tilde{a}(x_1),\tilde{a}(x_2),\tilde{a}(x_3)) = 0.$$

By squaring and adding this assignment is a zero of the entire system iff

$$\sum_{(x_1,x_2,x_3)\in\{0,1,\ldots,h\}^{3k}} \sum_{i=1}^{m} \chi_i(x_1,x_2,x_3)\cdot(p_i(\tilde{a}(x_1),\tilde{a}(x_2),\tilde{a}(x_3))^2 = 0. \tag{5.9}$$

Define $\tilde{g} : F^{3k} \to \mathbb{R}$ as

$$\tilde{g}(x_1, x_2, x_3) = \sum_{i=1}^{m} \chi_i(x_1, x_2, x_3) \cdot (p_i(\tilde{a}(x_1), \tilde{a}(x_2), \tilde{a}(x_3))^2 .$$

For (x_1, x_2, x_3) that are not indexes of variables χ_i is defined to be a trigonometric polynomial of degree $\log n$ extending the definition of χ_i given above. Since $\tilde{a}$ is a trigonometric max-degree $\log n$ polynomial, the p_i are algebraic max-degree 2 polynomials and the χ_i are trigonometric max-degree $\log n$ polynomials, it follows that $\tilde{g}$ is a trigonometric max-degree $\log n + 2 \cdot 2 \log n = 5 \log n$ polynomial. We define g in the same way as $\tilde{g}$ except that we replace $\tilde{a}$ by a. Since a is close to $\tilde{a}$ it follows that g is close to $\tilde{g}$. Thus, we finally can apply the sum-test procedure behind Theorem 5.8 and obtain a segmented $(\log(n), \text{polylog}(n))$-restricted test for equation (5.9). The low-degree test that the verifier performed is also segmented and $(\log(n), \text{polylog}(n))$-restricted. Altogether, a segmented and $(\log(n), \text{polylog}(n))$-restricted verifier for QPS has been constructed. □

5.5 Composition

One can check that the segments in a proof π for the QPS verification algorithm which we constructed in the previous section have length 1 or $O(\text{polylog}(n))$ and the verification can be decomposed into a series of tests that each consists of a check that a linear combination of a segment of length $O(\text{polylog}(n))$ equals the value in a segment of length 1 (this linear combination each time has the meaning of evaluating a polynomial in a certain argument, i.e., taking a certain linear combination of its coefficients).

A segment of length $O(\text{polylog}(n))$ encodes a univariate or bivariate trigonometric polynomial by specifying its coefficients. The domain on which the verifier may want to know the value of such a polynomial is F or F^2 where F is one of the finite fields used above to encode certain objects. One can verify that every such finite field has size $O(\text{polylog}(n))$ and thus the domains on which the verifier may want to evaluate the polynomial also have size $O(\text{polylog}(n))$. Let us mention again that we see such an evaluation at a certain point in the domain as taking a specific linear combination of the coefficients of the polynomial (i.e. the entries in a segment of length $O(\text{polylog}(n))$).

Hence, for every segment of length $O(\text{polylog}(n))$ there are $O(\text{polylog}(n))$ fixed linear combinations of the entries in the segment which the verifier may want to know. We now want to use verifier composition to enable the verifier to obtain the values of the linear combinations it wants to know without having to query every element in the segment $O(\text{polylog}(n))$ (we want it to evaluate a polynomial without querying every coefficient).

For this purpose we construct a QPS instance, so that we can use the same verification procedure within the verifier itself. We consider the coefficients of the polynomial as variables and for every linear combination that the verifier may want to know we introduce a result variable. That gives us one equation for every point in the domain where the verifier may want to evaluate the polynomial. Hence, we have $O(\text{polylog}(n))$ linear equations in total containing $O(\text{polylog}(n))$ variables.

Let us give an example to make things more clear. For simplicity we will consider an algebraic polynomial here. Suppose we have a segment specifying an algebraic polynomial of degree 2, then this segment must have length 3 and we consider its entries as variables c_0, c_1, c_2. Suppose that we want to evaluate this polynomial over the five element field. Then the verifier may want to know the following linear combinations of c_0, c_1, c_2: c_0, $c_2 + c_1 + c_0$, $2^2c_2 + 2c_1 + c_0$, $3^2c_2 + 3c_1 + c_0$ and $4^2c_2 + 4c_1 + c_0$. We introduce result variables y_0, y_1, y_2, y_3, y_4 and obtain the equations $i^2c_2 + ic_1 + c_0 = y_i$ for $i = 0, 1, 2, 3, 4$. This is a QPS instance (a QPS instance does not *have* to contain quadratic equations).

The QPS instance we have constructed is not yet in the right form for the verification procedure to be able to handle it. However, by introducing linearly many new variables and equations it can be transformed into an equivalent system of the right form in polynomial time. Note that for the construction of such a suitable QPS system the verifier does not yet need any information from the proof string.

The next step is to let the verifier expect the proof string to contain the values of all the result variables, so that in order to evaluate a polynomial only one query is needed. In order for this to work it is necessary that the verifier also checks consistency of the result variables (they should correspond to a polynomial of appropriate degree). To do this check the same verification procedure is used for the QPS instance constructed above. So, by querying a constant number of segments of length polyloglog(n) it can be checked whether this QPS instance has a solution. This question is of course not interesting because it basically asks whether there exists a polynomial of a certain degree (which is of course true). What makes this useful, is that the verification procedure not only checks that there is a solution, it also forces the verifier to specify this solution (encoded as a low degree polynomial). This specification of course also contains the values assigned to the result variables (and now this must be consistent or the verification procedure will likely reject). Using the correctness test the verifier can now obtain the correct value of any result variable using a constant number of queries only.

Let us now look at the complexity of this construction in terms of how many random bits and how many queries are necessary. We will refer to the verifier and the proof string before this construction as the old verifier and the old proof string and we will refer to the verifier and the proof string after this construction as the new verifier and the new proof string. The old proof string contains poly(n) many segments of length $O(\text{polylog}(n))$ to which we apply the above construction. This results in poly(n) many QPS instances of size $O(\text{polylog}(n))$. The lenth of a proof string for one such a QPS instance of size $O(\text{polylog}(n))$ is polynomial in polylog(n) and thus is still of order polylog(n). Polynomially many of such proof strings result in a new proof string which still is polynomial in n.

For the new verifier we have the following analysis. To decide which polynomials it wants to evaluate at which arguments the same procedure is used as the old verifier uses. Hence, this requires $O(\log n)$ random bits. For each polynomial it wants to evaluate (which are constantly many) it verifies the corresponding QPS instance of size $O(\text{polylog}(n))$. This verification needs $O(\log(\text{polylog}(n)))$ random bits and needs to query constantly

many segments of length $O(\text{polylog}(\text{polylog}(n)))$. Then, for each polynomial it wants to evaluate it needs to obtain (with high probability) the correct value of one of the result variables using the correctness test. This takes again $O(\log(\text{polylog}(n)))$ random bits and $O(\text{polylog}(\text{polylog}(n)))$ queries. Since the new verifier performs more tests the probability that the new proof string manages to mislead the verifier somewhere increases, but this can be compensated by repeating those tests a few more times because that only requires a constant factor of extra random bits and queries. Altogether this gives us a $(\log n, \text{polyloglog}(n))$-restricted verifier.

In the Arora et al. proof a segmented verifier with such resource bounds can be composed with a $(\text{poly}(n), 1)$-restricted verifier to obtain a $(\log(n), 1)$-restricted verifier. Since in the BSS model the analog of the classical $(\text{poly}(n), 1)$-restricted verifier is only $(\exp(n), 1)$-restricted, we need to perform the above construction yet another time in order to make the segments small enough such that even an exponential number of random bits relative to the lengh of the segment is still logarithmic in n.

Another such composition as we did above yields a $(\log n, \text{polylogloglog}(n))$-restricted verifier. The analysis is similar. Again using a same analysis, we can compose this verifier with the $(\exp(n), 1)$-restricted verifier to obtain the required $(\log(n), 1)$-restricted verifier. The main difference in this composition is that this time the QPS instances representing the polynomials are not encoded as low degree polynomials but as linear functions. The main point is that with these linear functions it is also possible to obtain with high probability the value of any of the result variables using only a constant number of queries. This completes the proof of the $\text{PCP}_{\mathbb{R}}$ theorem.

Chapter 6

Real PCP Theorem II: Proof via Gap Amplification

In the previous chapters we proved the $\mathrm{PCP}_{\mathbb{R}}$ theorem algebraically along the lines of Arora et al. and in this chapter we will show how it can also be proved combinatorially along the lines of Dinur's proof.

Dinur's proof is based on constructing a certain reduction between 3-SAT formulas in which for unsatisfiable formulas a so called gap amplification is obtained. The existence of such a reduction is known to imply the PCP theorem as we showed in Example 2.17. Nevertheless, at a first glance it is not clear whether similar ideas could be used for an appropriate $\mathrm{NP}_{\mathbb{R}}$-complete problem over the reals. The reason is that Dinur's proof heavily works with constraint systems that are to be solved over different finite alphabets as domains. The proof constructs several transformations between such finite alphabet constraint problems. It seems unclear whether the same can be done over uncountable structures. However, it turns out that this part de facto depends more on the combinatorial structure of the constraints involved than on the question over which domains they are to be solved. To see this, a view of real polynomial systems – the main objects involved in real number computations – is taken that seems a bit uncommon in algorithmic semi-algebraic geometry. The latter requires not only to consider the semi-algebraic solution set of such a system (as it is usually done in algebraic geometry), but to put more focus on a suitable grouping of the polynomials involved in the system. Then, it is more important to argue about common semi-algebraic solutions of some of these groups than of the entire system. It turns out that this can be accomplished as well over $\mathbb{R}$. Consequently, Dinur's proof seems less difficult to adapt to real computational models than the classical one. The arguments given hold as well for the complex number BSS model and its corresponding $\mathrm{PCP}_{\mathbb{C}}$ classes and $\mathrm{NP}_{\mathbb{C}}$. Since all required changes are minor or straightforward below we only add short remarks on the complex model where appropriate.

The chapter is organized as follows. Section 6.2 introduces the central $\mathrm{NP}_{\mathbb{R}}$-complete problem to be studied. It is a particular form of deciding solvability of polynomial systems; the problem is defined in such a way that the existence of a reduction amplifying the

unsatisfiability gap will imply the $\mathrm{PCP}_{\mathbb{R}}$ theorem. The construction of this reduction is given in Section 6.3.

6.1 Basic notions

We introduce the central decision problem to consider in this chapter. It deals with polynomial systems and is a variant of the Hilbert Nullstellensatz problem. However, the viewpoint under which we structure such systems is a bit unusual, so the definition at a first glance might look confusing. First, instead of considering a polynomial system as collection of single polynomial equations we group several polynomial equations together in *constraints*. Next, though each constraint still depends on real variables, we also group variables together in pairwise disjoint subsets, also called *arrays* below. This viewpoint allows to introduce two most important additional parameters: The number s of single variables that constitute a subset, and the number q of such subsets on which all polynomials in a constraint depend at most. Let us now formally define the decision problem, a clarifying example then follows after the definition.

Definition 6.1. *a) Let S be a finite set and $s \leq |S|$ be an integer. An s-partition of S is a partition of S into subsets of cardinality at most s. We call the subsets of an s-partition* arrays.

b) Let m, k, q, s be integers. An instance of the quadratic polynomial systems decision problem $QPS(m, k, q, s)$ is a collection $\{C_1, \ldots, C_m\}$ satisfying the following:

- *each C_i is a set of at most k quadratic real polynomials;*
- *let S denote the set of all variables the polynomials of the instance depend on. Then there exist an s-partition of S and for each C_i at most q many arrays in the partition such that all polynomials in C_i only depend on variables occurring in those q arrays.*

  Thus, each polynomial is a function $p : \mathbb{R}^{qs} \mapsto \mathbb{R}$.

Each C_i is called a constraint.

c) A constraint is satisfied *by a point $x \in \mathbb{R}^{qs}$ if x is a common zero of all polynomials in the constraint. The $QPS(m, k, q, s)$-instance is* solvable *if there is a common real solution for all its constraints.*

d) If above coefficients belong to $\mathbb{C}$ and variables range over $\mathbb{C}$ we obtain the complex $QPS(m, k, q, s)$ problem.

Remark 6.2. *a) Note that in principle there can be several constraints depending on the same set of arrays. This can lead to multiple edges between two vertices in the constraint graphs that we define below as multigraphs.*

b) To avoid a notational overhead if an array contains less than s variables we just add dummy variables that do not occur in the equations. Obviously, this does not influence solvability.

c) Below, we usually consider the parameters k, q, s as constants, whereas m is depending on the actual instance. In that sense it would be more correct to talk about $QPS(k, q, s)$-instances; however, at many places we argue about the changes in the number of constraints, so the above notation is intended to increase readability.

d) Over the reals parameter k in principle is not necessary. Here, using basic arguments such as introducing new variables for powers of variables, squaring and adding polynomials we could always choose $k = 1$ and the corresponding constraint to be given by a single polynomial equation $f(x) = 0$ of degree at most 4 with f being non-negative on a corresponding $\mathbb{R}^{qs}$, compare [15]. However, we notationally prefer to take care of k. First, in order to deal as well with the complex BSS model in which the above simplification does not work. Secondly, in many cases below we believe specifying k increases readability to state explicitly which polynomial equations enter into which constraints and might cause its unsatisfiability.

e) The quantity mqs is an upper bound for the number of variables an instance depends on.

Example 6.3. *This example is important throughout the rest of the chapter. We know that the standard QPS problem (Definition 2.13) is $\mathrm{NP}_{\mathbb{R}}$-complete [14]. We give two formulations of the QPS problem in the new framework by specifying different choices of parameters. The examples show $\mathrm{NP}_{\mathbb{R}}$-completeness of the QPS problem for the given choices of $k, q,$ and s.*

i) The standard QPS problem can be formulated as an instance in $QPS(m, 1, 3, 1)$. Each constraint consists of a single polynomial p_i, thus $k = 1$; the variable arrays all have dimension $s = 1$ and each polynomial depends on at most 3 such arrays.

ii) The problem $QPS(\tilde{m}, 1, 2, 3)$ in which each constraint of a problem instance has a single polynomial equation that depends on at most two variable arrays of dimension 3 is $\mathrm{NP}_{\mathbb{R}}$-complete. To see why we reduce $QPS(m, 1, 3, 1)$ to $QPS(\tilde{m}, 1, 2, 3)$. Given a $QPS(m, 1, 3, 1)$ instance we first consider the three variables of a single polynomial as different from those occurring in other polynomials. This will give the arrays of dimension 3 of the new instance. Then additional constraints have to be added in order to guarantee consistency between the above introduced copies of those variables that occurred in different constraints of the original system. Such a constraint consists of a single polynomial equation depending on two arrays. It claims equality between those variables in different arrays that have to be the same in the given system. We thus obtain a $QPS(\tilde{m}, 1, 2, 3)$ instance, where $\tilde{m}$ is a bound for m plus the number of different pairs of variable arrays.

Note that the above instances are equivalent to the standard QPS problem that we started with as far as solvability is concerned. Below it will be very important to argue about the number of constraints not satisfied if a system is unsolvable. For these arguments it is crucial to group the single polynomials into constraints.

For what follows QPS-instances with parameter $q = 2$ are most important. This gives the possibility of attaching canonically a constraint graph to such an instance. In this graph, vertices represent the variable arrays of the instance. If two arrays occur in the same constraint they are joined by an edge in the graph. It is then at many places important to study the structure of this graph and manipulate the underlying system in such a way that the resulting graph has nice properties.

The starting point of Dinur's proof is a simple observation which implies the PCP theorem if the existence of a very particular reduction can be established. We next recall this type of reduction and show the corresponding easy lemma for QPS and the real $\mathrm{PCP}_{\mathbb{R}}$ theorem.

Definition 6.4. *a) For a QPS(m, k, q, s)-instance ϕ denote by $UNSAT(\phi)$ the fraction of constraints in ϕ that cannot be satisfied in common, i.e.,*

$$UNSAT(\phi) := \frac{1}{m} \cdot \min_{x \in \mathbb{R}^{qs}} |\{i | x \text{ does not satisfy } i \text{ constraints in } \phi\}|.$$

b) A gap reduction *for QPS-instances is a polynomial time BSS algorithm having the following properties. There is a fixed $\epsilon > 0$ such that given a QPS(m, k, q, s) instance ϕ the algorithm computes an instance ψ in QPS(m', k, q, s) satisfying the following:*

i) if ϕ is satisfiable so is ψ;

ii) if ϕ is not satisfiable then $UNSAT(\psi) \geq \epsilon$.

Here, m' is polynomially bounded in m.

If an instance ϕ with m constraints is satisfiable $UNSAT(\phi) = 0$, otherwise $UNSAT(\phi) \geq \frac{1}{m}$. A gap reduction now amplifies for an unsatisfiable ϕ the ratio of unsatisfiable constraints from the trivial $1/m$ to a fixed ratio ϵ which is independent of the concrete instance. The following easy lemma is the basis of what has to be shown in the main section.

Lemma 6.5. *Suppose there exists a gap reduction for QPS with a fixed $\epsilon > 0$. Then the $\mathrm{PCP}_{\mathbb{R}}$-Theorem follows. The same holds over $\mathbb{C}$.*

Proof. Consider $\mathrm{QPS}(m, k, q, s)$ with a choice of parameters k, q, s that makes the problem $\mathrm{NP}_{\mathbb{R}}$-complete. Assuming the existence of a gap reduction a verifier works as follows on an instance $\phi \in \mathrm{QPS}(m, k, q, s)$. It first computes the formula ψ as result of the gap reduction applied to ϕ. As proof the verifier expects a satisfying assignment to the real variables of ψ. Next, it uniformly chooses a random constraint of ψ using the random bits and evaluates that constraint in the at most qs many coordinates of the assignment. The verifier accepts if the constraint is satisfied. Clearly, if ϕ – and thus also ψ – are satisfiable there is an assignment such that the verifier always accepts. If ϕ is not satisfiable, then with probability $\geq \epsilon$ the randomly chosen constraint is not satisfied by the assignment. Repeating the process finitely many times with the same assignment but independently chosen random constraints the verifier rejects with sufficiently high probability.

The arguments for the complex model are precisely the same. □

Our goal thus is to prove the existence of such a gap reduction.

6.2 The main proof

We shall now turn to the main part of the proof, the construction of a gap reduction for the $\mathrm{NP}_{\mathbb{R}}$-complete problem $\mathrm{QPS}(m, C, Q, 1)$, see Example 6.3. The parameters $C \geq 1$ and $Q \geq 3$ are constants that will be specified later. The structure of the proof is similar to that of the classical PCP theorem by Dinur. Its basic idea is as follows. Starting from a $\mathrm{QPS}(m, C, Q, 1)$-instance ϕ which is unsatisfiable an amplification step is performed. It constructs in polynomial time another QPS-instance ψ out of ϕ that has an increased unsatisfiability ratio. More precisely, if $UNSAT(\phi) = \epsilon$, then $UNSAT(\psi) \geq c \cdot \epsilon$ for a suitable constant $c > 1$ and ϵ small enough. Now in principle starting with $UNSAT(\phi) \geq \frac{1}{m}$ and repeating this amplification $\log m$ times the gap is increased from at least one unsatisfied constraint in ϕ to a constant fraction of unsatisfied constraints in the finally resulting instance. However, the amplification step increases the dimension of the variable arrays too much. Thus, before repeating amplification a *dimension reduction* step is performed that first reduces the parameter s again. Note that dimension reduction in Dinur's proof is called alphabet reduction. Over the reals, however, parameter s refers to the dimension of variable arrays, whereas the underlying alphabet is always infinite. We thus consider the changed notion to be more appropriate here.

Amplification is performed on instances having particularly structured constraint graphs. These graphs are so called expanders. Therefore, in a preprocessing step it has to be shown why it is possible to start with such particular instances.

The section is organized as follows. The first subsection relates the notion of expander graphs (see Section 2.4) to particular QPS instances. Next, we describe the preprocessing step used to put instances into the necessary format for amplification. It closely follows the classical preprocessing step, see [1]. The amplification step is described in the third subsection. Though basically Dinur's idea works over the reals as well, a lot of small details and calculations have to be changed. We thus include the full proof, always pointing out where differences to the discrete setting occur. The final step, dimension reduction, is given in subsection 6.2.4. It relies on the long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$ from (Chapter 3).

6.2.1 Nice QPS instances

In order to apply below the main steps necessary to establish the existence of a gap reduction for QPS, namely amplification and dimension reduction, we first have to preprocess a given instance. The goal of this preprocessing step is to obtain a QPS instance that has a constraint set which in a certain sense is highly structured. Such instances will be called *nice*. Niceness is modelled using expanders, a well known concept from graph theory. Throughout this section we consider QPS instances whose constraints depend on

two variable arrays, i.e., for which parameter $q = 2$. This allows to canonically attach a constraint graph to the instance.

Definition 6.6. *Let ϕ be a QPS$(m, k, 2, s)$-instance, $m, k, s \in \mathbb{N}$. The* constraint graph *$G = (V, E)$ attached to ϕ is an undirected multigraph with one vertex for each variable array of ϕ. For two different such arrays i, j there is an edge $(i, j) \in E$ iff ϕ has a constraint that depends on the arrays i and j. Different constraints depending on the same arrays result in multiple edges. And there is a loop $(i, i) \in E$ for an array $i \in V$ iff ϕ has a constraint that only depends on that single array i. Again, several such constraints result in multiple loops.*

By a little abuse of notation below we sometimes denote an edge e by the vertices it connects $e = (i, j)$ even though there might be multi-edges connecting the same i and j.

General assumption: Unless otherwise stated all graphs below are supposed to be undirected multigraphs that can have loops.

Before the amplification step is performed it is necessary to guarantee that this constraint graph has a particular nice structure.

We use the graph theoretical definitions in section 2.4 to define the concept of a nice instance. In the next subsection it is then shown why without loss of generality we can start from a nice QPS instance.

Let us start with a remark about how we treat loops in the random walk matrix. At a first glance a loop might be considered as one edge and one constraint. Due to the application of constraint graphs we have in mind for the amplification step this will be done a bit differently. Our main purpose when considering random walks on a constraint graph is to guarantee that all edges occur with the same probability as, say, first edge of such a walk if a vertex is chosen at random. To achieve this property we consider loops as contributing one edge which, however, in a random walk is chosen with twice the probability of edges that are no loops. There is still one constraint attached to a loop, and the loop contributes 2 to the degree d of its vertex. Consequently, for the random walk matrix a loop contributes $\frac{2}{d}$ to the corresponding diagonal entry.

The theory of expander graphs now is used in order to specify those QPS$(m, k, 2, s)$-instances we are interested in. More precisely, we want to assure that the constraint graph corresponding to such an instance has to be an expander graph satisfying certain additional properties.

Definition 6.7. *Let $d \in \mathbb{N}$ be a fixed constant to be specified below. A QPS$(m, k, 2, s)$-instance ϕ is called* nice *if the following conditions hold:*

i) the constraint graph of ϕ is a d-regular expander with expansion parameter $\lambda(G) \leq 0.9$.

ii) for each vertex of the graph at least one third of the edges incident to that vertex are loops.

Our first goal is to justify why it is no restriction requiring that QPS instances have to be nice. Towards this aim constant $d \in \mathbb{N}$ in the above definition will later be chosen according to Theorem 2.23 for a suitable value of λ.

6.2.2 Preprocessing

Before the amplification step is applied in the main proof the QPS-instances must have a normalized form, i.e., they have to be nice. The preprocessing step achieves this goal in several small steps. The following idea is central to put more structure onto an instance. Starting from a system of constraints each depending on at most two variable arrays (so that we have a constraint graph) one first replaces each occurrence of such an array in different constraints by an own new copy of the array. This of course decouples occurrences of the same array in several constraints. In order to repair that effect new constraints are added which relate the copies among each other. Instead of doing this in a straightforward manner (as one would be tempted if just considering the semi-algebraic solution set) it is done putting more structure onto the constraints. More structure here means to take care about the fraction of non-satisfied constraints in case the original system is not satisfiable. This additional structure is obtained by using expander graphs for those constraints relating the copies of the same array.

Before this idea is realized it is first shown how to efficiently transform $\mathrm{QPS}(m, k, q, s)$-instances for arbitrary values of q into instances where $q = 2$. We have already noted above that solvability of $\mathrm{QPS}(m, k, 2, s)$ instances is $\mathrm{NP}_{\mathbb{R}}$-complete for suitable choices of s. However, in the course of the dimension reduction step instances are blown up with respect to parameter q and then have to be transformed back to get $q = 2$. The first lemma shows how to easily reduce q again.

Lemma 6.8. *Let ϕ be a $QPS(m, k, q, s)$ instance where $q > 2$. Then one can in polynomial time compute an instance $\psi \in QPS(qm, k + s, 2, qs)$ such that*

- *if ϕ is satisfiable so is ψ and*
- *if $UNSAT(\phi) \geq \epsilon > 0$, then $UNSAT(\psi) \geq \frac{\epsilon}{q}$.*

Proof. Let $x_i, 1 \leq i \leq n$ denote the variable arrays occurring in ϕ and taking values in $\mathbb{R}^s$. In addition to the x_i the new instance depends on m further arrays $y_j, 1 \leq j \leq m$, one for each old constraint. The y_j range over $\mathbb{R}^{qs}$ and are supposed to equal the q many arrays among the x_i that occur in constraint C_j. Now each such constraint is replaced by q many new ones in the following way: if x_i is an array occurring in C_j, then a new constraint C_{ij} is defined by s equations of degree 1 which state equality between the coordinates of x_i and the corresponding s coordinates of y_j. In addition, C_{ij} requires the old constraint C_j to be fulfilled, but stated using the single new variable array y_j instead of the q old arrays among $\{x_1, \ldots, x_n\}$.

This way we obtain qm many constraints, each depending on one y_j and one x_i, having variable arrays of dimension at most qs, and adding s polynomial equations to

the $\leq k$ many in the given instance. It is obvious that ψ is satisfiable if ϕ is. Assume then that each assignment to the variables of ϕ has to violate ϵm many constraints. Let $(y, x) \in \mathbb{R}^{mqs} \times \mathbb{R}^{ns}$ be an assignment for ψ. Then x as assignment for ϕ has to violate at least ϵm constraints. Let C_j be one of those. We show that (y, x) violates at least one of the constraints C_{ij}. For if y_j equals the arrays among x that occur in C_j then all C_{ij} are violated. And if at least one inconsistency occurs between y_j and an x_i, then this particular C_{ij} is violated. Altogether (y, x) violates at least ϵm many constraints which results in $UNSAT(\psi) \geq \frac{\epsilon}{q}$. □

The next preprocessing step puts more structure on the constraint graph. Fix a number $0 < \rho \leq 0.1$ and define $d := d(\rho) \in \mathbb{N}$ such that according to Theorem 2.23 there is an efficiently computable family of d-regular edge-expander graphs with expansion rate ρ.

Lemma 6.9. *Let d be as above, let a $QPS(m, k, 2, s)$-instance ϕ be given, $m, k, s \in \mathbb{N}$; let $C_1, \ldots, C_m$ denote the set of constraints in ϕ and $x_1, \ldots, x_n$ the variable arrays ranging over $\mathbb{R}^s$. We can construct another instance $\psi \in QPS((d+1) \cdot m, \max\{k, s\}, 2, s)$ in polynomial time such that the following holds:*

i) the constraint graph of ψ is $d+1$-regular;

ii) if ϕ is satisfiable so is ψ;

iii) if ϕ is not satisfiable with a fraction of at least $\epsilon > 0$ unsatisfiable constraints, then ψ is unsatisfiable with a fraction of at least $c \cdot \epsilon$ unsatisfied constraints. Here, c is a constant satisfying $c > \frac{1}{80(d+1)}$.

Proof. The construction of the new instance ψ is done in two steps. First, replace each occurrence of a variable array x_i in a constraint of ϕ by an own new copy; if x_i occurs in $l_i \leq m$ constraints of ϕ we thus obtain l_i new variable arrays $y_i^{(1)}, \ldots, y_i^{(l_i)}$ all of dimension s. The m old constraints C_j are maintained, but now depending on the corresponding copies of the variable arrays involved. This gives again m constraints in ψ and each new variable array occurs in exactly one of them. Because $\sum_{i=1}^{n} l_i \leq 2m$ there are at most $2m$ variable arrays in ψ.

Next, consistency among the copies of an old variable array x is expressed using expander graphs. Let $\{G_n\}_{n \in \mathbb{N}}$ be the d-regular explicit edge expander graph family with expansion rate $\rho \geq 0.1$ whose existence was guaranteed when we fixed the constant d. Let x denote a variable array in ϕ and $y_1, \ldots, y_\ell$ the corresponding variable arrays in ψ as described above. Consistency between these copies is expressed via constraints arising from the expander G_ℓ in the following way. To each edge in G_ℓ connecting an array $y_j = (y_{j1}, \ldots, y_{js})$ and an array $y_k = (y_{k1}, \ldots, y_{ks})$ we attach as constraint the s linear equations $y_{jt} - y_{kt} = 0$, $t = 1, 2, \ldots, s$. The constraint thus claims equality of the two copies.

This way the new instance ψ has at most $2m$ variable arrays, each occurs in $d+1$ many constraints and the total number of constraints is $m + \frac{1}{2} \cdot \sum_{i=1}^{n} l_i d \leq (d+1)m$.

It is clear that if ϕ is satisfiable so is ψ; a satisfying assignment for the former instance canonically gives one for the latter where all copies get the same correct assignment. It remains to show the lower bound for $UNSAT(\psi)$ if ϕ is not satisfiable. Let $\tilde{y} \in \mathbb{R}^{mns}$ denote an arbitrary assignment to the variables of ψ. To $\tilde{y}$ there naturally corresponds the following *plurality assignment* $\tilde{x} \in \mathbb{R}^{ns}$ for the variables in ϕ: for a variable array x_i choose as assignment the point in $\mathbb{R}^s$ that most frequently occurs as assignment for the l_i copies of x_i breaking ties arbitrarily. Let t_i denote the number of copies $y_i^{(j)}$ that do not get this plurality assignment under assignment $\tilde{y}$; trivially, $t_i \leq l_i - 1$.

Since ϕ is assumed to be unsatisfiable the assignment $\tilde{x}$ violates at least ϵm many of the constraints $C_1, \ldots, C_m$, where $UNSAT(\phi) := \epsilon > 0$. Depending on how large the sum $\sum_{i=1}^{n} t_i$ is we either get the required number of violated constraints in ψ among the $C_1, \ldots, C_m$ or among the constraints arising from the expander graphs. More precisely, if $\sum_{i=1}^{n} t_i \leq \frac{\epsilon m}{4}$, then non-consistent assignments in $\tilde{y}$ for copies of the same array x_i can contribute at most for $\frac{\epsilon m}{4}$ of the ϵm many violated constraints among $C_1, \ldots, C_m$. Therefore, $\tilde{y}$ violates at least $\frac{3}{4}\epsilon m$ constraints among the original constraints of ϕ. In this case, $UNSAT(\psi) \geq \frac{3\epsilon}{4(d+1)}$. On the other hand assume $\sum_{i=1}^{n} t_i > \frac{\epsilon m}{4}$. The required number of unsatisfied constraints is now obtained among those which arise from the expanders used. For a fixed $1 \leq i \leq n$ let S denote a set of vertices in G_{ℓ_i} that do get the same assignment under $\tilde{y}$, but not the plurality assignment. Obviously, $|S| \leq \frac{n}{2}$ and the sum of the cardinalities of all such sets S equals t_i. By the fact that G_{ℓ_i} is an edge-expander with rate ρ we therefore get at least $\frac{1}{2}\rho d t_i$ many unsatisfied consistency constraints for each i, since at least that many edges in G_{ℓ_i} connect copies which get different assignments under $\tilde{y}$. It follows $UNSAT(\psi) \geq \frac{d}{80(d+1)}\epsilon$. Altogether this results in $UNSAT(\psi) \geq c\epsilon$, where $c := \min\{\frac{3}{4(d+1)}, \frac{d}{80(d+1)}\} \geq \frac{1}{80(d+1)}$. □

The only thing left to do with respect to preprocessing is to add some empty constraints (constraints that are always true) in order to make the constraint graph nice, i.e., an expander with sufficiently many loops. For the lemma below remember the definition of the random walk matrix (Definition 2.19).

Lemma 6.10. *There is a constant $d \in \mathbb{N}$ together with a polynomial time computable reduction from QPS-instances to QPS-instances such that the following holds: Let $d' \leq d$ and ϕ be a QPS$(m, k, 2, s)$-instance with a constraint graph that is d'-regular. The reduction computes a QPS$(3dm, k, 2, s)$-instance $\psi = \psi(\phi)$ such that*

- *the constraint graph of ψ is a $4d$-regular expander with an algebraic expansion parameter $\lambda \leq 0.9$;*
- *for every vertex in the constraint graph of ψ at least one third of the edges that are incident to this vertex are loops [1];*
- *if ϕ is satisfiable, then ψ is satisfiable and*
- *if ϕ is not satisfiable, then UNSAT$(\psi) \geq$ UNSAT$(\phi)/(4d)$.*

Proof. The construction of ψ works in several steps. Let d be such that there exists a family $\{G_n\}_n$ of d-regular expander graphs with (algebraic) expansion parameter ≤ 0.6; its existence is guaranteed by Theorem 2.23.

The new instance depends on the same variable arrays as the given ϕ. We start with including the original constraints as well for ψ. Next, add empty loop-constraints to each vertex such that the new constraint graph becomes d-regular. Here, 'empty' means that the constraint is always satisfied. Since $1 \leq d' \leq d$ this will reduce the ratio of unsatisfied constraints at most by a factor $\frac{1}{d}$.

Denote the corresponding constraint graph by H_1 and let n be its number of vertices. Next, we add constraints to H_1 exploiting the expander structure of G_n. More precisely, for each edge in G_n an empty constraint between the corresponding vertices is added to H_1. The new constraint graph H_2 is $2d$-regular and the unsatisfiability factor is reduced by a factor $\frac{1}{2}$. Finally, add d many loops (thus increasing the degree by $2d$ and reducing the unsatisfiability factor by factor $\frac{1}{2}$) as empty constraints to each of the n vertices. We end up with a $4d$-regular constraint graph G_ψ for ψ that satisfies the condition on the number of loops. It remains to show that G_ψ is as well an expander with expansion parameter ≤ 0.9.

Let A_ψ denote the corresponding random walk square matrix. Using Definition 2.19 it is easy to see that A_ψ can be decomposed into the sum $\frac{1}{4}A_{G_n} + \frac{1}{4}A_{H_1} + \frac{1}{2}Id$, where the first two matrices are those attached to the corresponding graphs and the (n, n)-identity matrix results from the final step of adding d loops. Sub-additivity of λ (see Lemma 2.24) as well as the fact $\lambda(H_1) \leq 1$ since H_1 is d-regular finally give $\lambda(G_\psi) \leq \frac{1}{4}\lambda(G_n) + \frac{3}{4} \leq 0.9$. □

Note that since above we used an expander family with $\lambda \leq 0.6$ by Theorem 2.22 there is a family of edge-expanders with regularity $d - 1$ and $\rho \geq 0.1$ that could be used in Lemma 6.9. Now putting the constructions of the above lemmas together we finally can conclude the following.

[1] according to Definition 2.19 this implies that for each step in a random walk the probability that a loop is taken is at least $\frac{1}{2}$

Theorem 6.11. *There exist a constant $d \in \mathbb{N}$ and a polytime computable function from QPS-instances to QPS instances which maps a QPS(m, k, q, s)-instance ϕ to a nice instance ψ in QPS$(3qd^2m, k + qs, 2, qs)$ such that*

- *if ϕ is satisfiable, then ψ is satisfiable.*
- *if ϕ is not satisfiable, then $UNSAT(\psi) \geq UNSAT(\phi)/(320qd^2)$.*

It is important for what follows that above both the number of constraints is increased and the unsatisfiability factor is decreased by a constant factor only.

6.2.3 Amplification

Given a nice QPS-instance the allover purpose now is to perform a logarithmic number of reduction rounds to increase the unsatisfiability gap. The first step in a round is an amplification step which increases the ratio of unsatisfied constraints by a constant factor > 1. After the amplification step a dimension reduction step reduces again the dimension of the variable arrays which has been increased during amplification. In this subsection amplification is explained. We first give a more informal outline of a single amplification step before proofs are presented. Once again, the structure of this section closely follows [1].

Suppose a nice QPS$(m, k, 2, s)$-instance ψ is given. Let d denote the corresponding regularity parameter and let $n \leq 2m$ denote the number of variable arrays. The goal is to construct an instance which either is satisfiable if ψ was or in which for any assignment a constant fraction $\epsilon_{final} > 0$ (to be specified) of the constraints will not be satisfied. We will concentrate our arguments on how amplification works for unsatisfiable instances ψ which have a gap that is too small, i.e., smaller than ϵ_{final}. If $UNSAT(\psi)$ is already large enough it will stay above this ϵ_{final} after the reduction, see below.

So we assume that the input instance has a gap which is smaller than some constant which we will specify later. Our goal is to construct in polynomial time an instance ψ^t in some QPS$(m(t), k(t), 2, s(t))$ such that $UNSAT(\psi^t) \geq c \cdot UNSAT(\psi)$ for a suitable constant $c > 1$. Here, $t \in \mathbb{N}$ is a constant that results from the construction.

Construction of ψ^t.

Let $t \in \mathbb{N}$ be fixed. We start with the explanation of how the variables and constraints of the new instance ψ^t are built. It is an adaptation of Dinur's corresponding construction to the real case.

Suppose instance ψ has n variable arrays $x_1, \ldots, x_n$, also called *old* arrays below. The x_i range over $\mathbb{R}^s$. The instance ψ^t to be constructed also depends on n *new* arrays denoted by $y_1, \ldots, y_n$ and ranging over $\mathbb{R}^{s(t)}$. Here, the new array dimension is $s(t) := d^{t+\sqrt{t}+1} \cdot s$. The variables of a new array are divided into blocks of s coordinates each. This is done in order to relate assignments for the new arrays with assignments for some of the old arrays as follows. For a vertex i in G_ψ consider the set $U_{t+\sqrt{t}}(i)$ of vertices in G_ψ that

can be reached from i by a walk of at most $t + \sqrt{t}$ many steps. Note that $i \in U_{t+\sqrt{t}}(i)$ and $U_{t+\sqrt{t}}(i)$ contains at most $d^{t+\sqrt{t}+1}$ many vertices due to d-regularity of G_ψ. Each of the $d^{t+\sqrt{t}+1}$ blocks of dimension s in array y_i is related to an old variable array x_j for a vertex $j \in U_{t+\sqrt{t}}(i)$. After it has been fixed how blocks of s consecutive coordinates in y_i are related to some old arrays x_j we can as well relate each assignment for $y_i \in \mathbb{R}^{s(t)}$ to an assignment $x_j \in \mathbb{R}^s$ for all $j \in U_{t+\sqrt{t}}(i)$.

Definition 6.12. *Let $y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \cdot s(t)}$ be an assignment for the new variables. If y_i and an x_j are related in the way described above we say that y_i* claims *its assignment for the old array x_j.*

Of course, if $j \in U_{t+\sqrt{t}}(i) \cap U_{t+\sqrt{t}}(i')$ for different $i \neq i'$, then assignments for $y_i, y_{i'}$ may claim different assignments for x_j.

Next, we describe how the new constraints in ψ^t are defined. The goal is to increase the unsatisfiability ratio of ψ. This is achieved by introducing relatively many constraints (the increase is linear, but with a high factor) which comprise a lot of redundancies.

Each constraint in ψ^t results from a walk in G_ψ of length $2t$, i.e., an alternating sequence $(i_1, e_1, i_2, e_2, \ldots, i_{2t}, e_{2t}, i_{2t+1})$ of vertices i_j and edges $e_j := (i_j, i_{j+1}), 1 \leq j \leq 2t$ in G_ψ. The constraint attached to this walk depends on the two new arrays y_{i_1} and y_{2t+1}. Its polynomial equations express the following two requirements:

1. Consistency-between-new-variables requirement: Since an assignment for y_{i_1} claims assignments for $x_{i_1}, x_{i_2}, \ldots, x_{i_{t+\sqrt{t}+1}}$ and an assignment for $y_{i_{2t+1}}$ claims assignments for $x_{i_{2t+1}}, x_{i_{2t}}, \ldots, x_{i_{t-\sqrt{t}+1}}$, the old variables x_{i_j} for $j \in \{t - \sqrt{t} + 1, \ldots, t + \sqrt{t} + 1\}$ receive assignments from both new arrays. We thus include for all those j linear equations expressing that y_{i_1} and $y_{i_{2t+1}}$ assign the same values on all the coordinates of such an x_j. This contributes $(2\sqrt{t} + 1) \cdot s$ linear equations to the constraint.[2]

2. Consistency-with-old-constraints requirement: As explained above for $j \in \{t - \sqrt{t} + 1, \ldots, t + \sqrt{t}\}$ and an edge $e_j = (i_j, i_{j+1})$ in the walk, the arrays x_{i_j} and $x_{i_{j+1}}$ inherit assignments from assignments to the arrays $y_{i_1}, y_{i_{2t+1}}$; to each of these $2\sqrt{t}$ many edges there corresponds an old constraint of ψ relating x_{i_j} and $x_{i_{j+1}}$. The new constraint requires as well that those old constraints are satisfied by the values claimed for the old variable arrays through y_{i_1} (or equivalently, because of item 1., through $y_{i_{2t+1}}$).

Requirement 2. for each j is the same as in the given instance just changing variables. So each constraint in ψ^t is made of $\leq k(t) := 2\sqrt{t}k + (2\sqrt{t} + 1) \cdot s$ many polynomial equations. Since G_ψ is regular there are at most $m(t) := n \cdot d^{2t}$ many walks of length $2t$, so this bounds as well the number of constraints in ψ^t.

[2] These requirements are not included in the classical construction due to the underlying finite alphabets. It will become obvious below why it is needed over the reals.

It is easy to see that a satisfying assignment for ψ extends to one for ψ^t. Just propagate the assignment to the y_i's according to the first consistency requirement above. The hard part to see is why an unsatisfiability ratio of a given (unsatisfiable) ψ is increased by the construction. Towards this aim we relate any assignment for all new variable arrays to a so-called plurality assignment for the old arrays. Assuming this plurality assignment (like any other) to violate a fraction of $UNSAT(\psi)$ many constraints in ψ it is then shown that the given assignment y for ψ^t violates a ratio of $\geq c \cdot UNSAT(\psi)$ many constraints of ψ^t. This reasoning closely follows Dinur's one. However, due to the fact that assignments stem from the uncountable set $\mathbb{R}^{n \cdot s(t)}$ instead of a finite set some arguments have to be adjusted. One necessary change in order to perform this adjustment is the inclusion of the consistency-between-new-variables requirement above.

The plurality assignment.

Given an assignment $y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \cdot s(t)}$ for the n variable arrays $y_i \in \mathbb{R}^{s(t)}$ of ψ^t, we first define the plurality assignment inferred from it to the old arrays $x_j \in \mathbb{R}^s$. Let $t \in \mathbb{N}$ be fixed. Consider a vertex i of G_ψ together with a random walk in G_ψ of length t starting in i. Remember the way loops are treated in such a walk, see the remarks on page 82. With a certain probability the walk reaches a vertex j which obviously belongs to the t-neighborhood $U_t(i)$ of i. Then the given assignment for y_j in particular assigns values to the coordinates of the old array x_i. We collect all the assignments to x_i that result from such random walks with the corresponding probabilities that a j is the end point of the walk and thus y_j infers its assignment to x_i.

Definition 6.13. *Given an assignment $y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \cdot s(t)}$ for the variables of ψ^t the* plurality assignment $x^{pa} := (x_1^{pa}, \ldots, x_n^{pa}) \in \mathbb{R}^{n \cdot s}$ *for the variables of ψ is defined as follows: For $1 \leq i \leq n$ we define x_i^{pa} as the assignment resulting with highest probability from y according to the above random walk process. Ties can be broken arbitrarily.*

One technical difference to the discrete setting has to be pointed out here. Over the reals it is only guaranteed that the plurality assignment for an array x_j occurs at least once. In contrast, if the variables take values from a finite alphabet there is a constant lower bound on the probability with which the plurality assignment occurs; this bound depends only on the alphabet size but not on t. This difference requires below a modification of the discrete arguments.

Outline how to amplify the unsatisfiability ratio.

Suppose then that ψ is unsatisfiable with $UNSAT(\psi) = \epsilon > 0$. Let an arbitrary assignment y for ψ^t and the related plurality assignment x^{pa} for ψ be fixed. Every assignment for the old variables violates $\geq m \cdot \epsilon$ constraints in ψ, where m denotes the number of constraints in ψ. Thus, in particular x^{pa} violates that many constraints. Our goal is to show that y violates at least a fraction of $\epsilon(t) = c \cdot \epsilon$ constraints in ψ^t for a large enough value $c > 1$.

This is achieved by analyzing two cases. Consider an edge $e = (i_j, i_{j+1})$ in G_ψ such that the plurality assignment $(x^{pa}_{i_j}, x^{pa}_{i_{j+1}})$ violates the corresponding constraint in ψ.

a) Suppose both the assignments $x^{pa}_{i_j}, x^{pa}_{i_{j+1}}$ have been obtained because for relatively many (to be specified) endpoints of the random walk construction the corresponding array of y infers this assignment. Then it is shown that many of the endpoints of $2t$-step walks of G_ψ in which e occurs in the middle segment result in the plurality values for $x^{pa}_{i_j}, x^{pa}_{i_{j+1}}$, i.e., y violates many constraints in ψ^t. This part is analyzed similarly as in the finite alphabet situation.

b) If at least one of the values $x^{pa}_{i_j}$ or $x^{pa}_{i_{j+1}}$ has been obtained by few endpoints only (but still represents the majority of the occurring values) we show that y violates the consistency-between-new-variables requirements in a lot of constraints. This case has to be handled because of the reals as underlying structure.

We now calculate a lower bound for the expectation of a random variable V that counts the number of edges with unsatisfied constraints in a random walk. We also calculate an upper bound for the square $E[V^2]$ of the number of such edges in a random walk. Since for any non-negative random variable V taking integral values by an application of the Chebychev-Cantelli inequality it is $Pr[V > 0] \geq E[V]^2/E[V^2]$ this will then give us a lower bound on the fraction of walks for which the corresponding constraint in ψ^t is not satisfied.

Analysis of Case a).

Assume we have an edge $e = (i_j, i_{j+1})$ such that the corresponding constraint in ψ is violated by the plurality assignment. We start by considering case a) and assume that the plurality values of $x_{i_j}, x_{i_{j+1}}$ are obtained relatively often. At first sight this information seems pretty useless because if we look at the set of walks in which e occurs in the middle segment then it is obvious that for almost all of them the distance (along the walk) from x_{i_j} and $x_{i_{j+1}}$ to the respective endpoints is not t. The plurality assignment was defined using random walks of length t, so it does not say anything directly about walks which have a different length. To solve this problem we need the many loops guaranteed to exist by the niceness condition. Their existence implies that a random walk of t steps statistically is not too different from a random walk which has a few steps more or less. If the random walk starts in u, the probability that we end in v only changes very slightly if we make our walk a few steps longer or shorter. The following lemma makes this precise.

Lemma 6.14. *Let $t \in \mathbb{N}, \delta \leq 1/160$ and $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$. If the plurality assignments for x_{i_j} and $x_{i_{j+1}}$ both occur with probability at least $\frac{5}{8}$, then the following holds. For a fraction of at least $\frac{1}{4}$ of the walks of length $2t$ that have $e = (i_j, i_{j+1})$ as j-th edge the values that the starting point y_{i_1} and the endpoint $y_{i_{2t+1}}$ of the walk claim for x_{i_j} and $x_{i_{j+1}}$, respectively, agree with the plurality assignments for those arrays.*

Proof. We view i_1 as a vertex reached after a random walk of $j-1$ steps from i_j and i_{2t+1} as a vertex reached after a random walk of $2t-j$ steps from i_{j+1}. Since these two random walks are independent the probability that y_{i_1} claims the plurality value for x_{i_j} and $y_{i_{2t+1}}$ claims the plurality value for $x_{i_{j+1}}$ is just the product of the two separate probabilities. We claim that both probabilities are at least $\frac{1}{2}$ and prove it for y_{i_1}.

Recall that at every node at least $\frac{1}{3}$ of the edges incident to it are loops. Moreover, the probability to choose a specific loop as next step in the walk is twice the probability to choose a non-loop edge. Thus a random walk can be modelled as follows. At each vertex we consider a subset of the loops incident to this vertex such that the probability to choose an edge from this set is exactly $\frac{1}{2}$. We call the chosen loops non-real edges. Such a set of loops exists because ψ is nice. The other edges incident to this vertex are called real edges. There can be loops among the real edges, but all non-real edges must be loops. Next a random walk is chosen as follows. At each step a fair coin is tossed. If the coin comes up head we choose a random edge among the real ones and if the coin comes up tail we choose a random edge among the non-real ones.

Let S_ℓ denote the random variable measuring how often a fair coin comes up head in ℓ tosses. For ℓ, ℓ' such that $\ell - \delta\sqrt{\ell} \leq \ell' \leq \ell + \delta\sqrt{\ell}$ it is folklore to show that variables S_ℓ and $S_{\ell'}$ are within statistical difference at most 20δ, i.e.,

$$\sum_{k=0}^{\max\{\ell,\ell'\}} |\Pr[S_\ell = k] - \Pr[S_{\ell'} = k]| \leq 20\delta$$

(see [1], pages 469 and 542). Now for a $(j-1)$-step walk starting in i_j and stopping in i_1 split the probability that y_{i_1} claims $x_{i_j}^{pa}$ according to the total number of real edges taken:

$$\Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j}] = \sum_{k=0}^{j-1} \Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j} | S_{j-1} = k] \cdot \Pr[S_{j-1} = k].$$

Let v be the endpoint of a random walk of length t out of i_j. By assumption

$$\Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j}] \geq \frac{5}{8}$$

and, as above,

$$\Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j}] = \sum_{k=0}^{t} \Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j} | S_t = k] \cdot \Pr[S_t = k].$$

Now S_t and S_{j-1} are statistically close because $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$.

This implies

$$\begin{aligned}
&\Big|\Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j}] - \Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j}]\Big| \leq \\
&\sum_k \Big|\Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j} | S_t = k] \cdot \Pr[S_t = k] - \\
&\qquad\qquad \Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_{i_j} | S_{j-1} = m] \cdot \Pr[S_{j-1} = m]\Big| \\
&\leq \sum_k |\Pr[S_t = k] - \Pr[S_{j-1} = k]| \leq 20\delta \leq \tfrac{1}{8}.
\end{aligned}$$

The second last inequality above follows because for all k considered it is

$$\Pr[y_v \text{ claims } x_{i_j}^{pa} \text{ for } x_j | S_t = k] = \Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_j | S_{j-1} = k] \leq 1.$$

Hence

$$\Pr[y_{i_1} \text{ claims } x_{i_j}^{pa} \text{ for } x_j] \geq \Pr[y_v \text{ assigns plurality value to } x_j] - \frac{1}{8} \geq \frac{1}{2}.$$

The same argument applies to $y_{i_{2t+1}}$ and $x_{i_{j+1}}$, thus finishing the proof. □

Analysis of Case b).

In order to obtain the desired lower bound for $E[V]$ next we have to deal with the case where the plurality assignment is claimed with a small probability only. The following shows that in this case the corresponding edge e leads in a large fraction of the walks to a violation of the corresponding constraint via case b).

Lemma 6.15. *Let $t \in \mathbb{N}, \delta \leq 1/160$ and $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$. If the plurality assignment for x_{i_j} occurs with probability less than $\frac{5}{8}$ the following holds. For a fraction of at least $\frac{1}{4}$ of the walks of length $2t$ that have e as j-th edge the values that the starting point y_{i_1} and the endpoint $y_{i_{2t+1}}$ of the walk claim for x_{i_j} disagree.*

The corresponding statement is true for $x_{i_{j+1}}$.

Proof. Let v once more denote the vertex reached after a random walk of length t out of i_j. Let $r \in \mathbb{R}^s$ denote an arbitrary assignment of dimension s. As in the previous proof it is

$$\Pr[y_{i_1} \text{ claims } r \text{ for } x_{i_j}] \leq \Pr[y_v \text{ claims } r \text{ for } x_{i_j}] + \frac{1}{8}.$$

By assumption, for every $r \in \mathbb{R}^s$ the probability that y_v assigns r to x_j is less than $\frac{5}{8}$. The probability that y_{i_1} assigns r to x_j therefore is less than $\frac{3}{4}$. Consequently, the probability that the assignments which y_{i_1} and $y_{i_{2t+1}}$ give to x_{i_j} disagree is at least $\frac{1}{4}$. □

Let F denote the set of edges in the instance ψ such that the corresponding constraint is violated by the plurality assignment. Recall that our goal is to prove a lower bound for $\Pr[V > 0]$, where V is the random variable which counts the number of edges e in

a random $2t$-step walk that satisfy the following: e belongs to F, it is the j-th edge in the walk for a $j \in \{t - \delta\sqrt{t}, \ldots, t + \delta\sqrt{t}\}$, where $\delta = 1/160$ and causes the corresponding constraint in ψ^t to be unsatisfied by assignment y. We are now able to extract such a lower bound.

Lemma 6.16. *Let $\epsilon \leq \frac{1}{d \cdot \sqrt{t}}$, let ψ be an instance with $UNSAT(\psi) = \epsilon$ and ψ^t be constructed as above. For the random variable V as defined above it is $Pr[V > 0] \geq c \cdot \epsilon$ with $c = \frac{\sqrt{t}}{3520d}$.*

Proof. The two previous lemmas show that $E[V] \geq \frac{1}{4} \cdot 2\delta\sqrt{t} \cdot \mathrm{UNSAT}(\phi) = \frac{1}{2}\delta\sqrt{t}\epsilon$. Since for the non-negative random variable V it is $Pr[V > 0] \geq \frac{E[V]^2}{E[V^2]}$ we shall find an upper bound for $E[V^2]$. Here the expander property of the constraint graph is used. Note that one can just consider edges $e \in F$ and disregard the additional requirement that the edges counted by V lead to a violation in the constraint corresponding to the walk. Disregarding this can only increase the value.

Let S be the set of vertices incident to F. Since d is the degree of the constraint graph there are $\frac{1}{2}d$ times as many edges as vertices. Every edge is incident with at most two vertices, hence $|S| \leq 2|F|$ and $|S|/|V| \leq d|F|/|E| = d\epsilon$.

In the calculation we shall need to know the probability of staying in S if we start from a random vertex in S and from this vertex take one random step in the graph. Since there are $|S|$ vertices which each have degree d, there are $d|S|$ possibilities that are equally likely. Theorem 2.22, part a) says that

$$|E(S, \bar{S})| \geq (1 - \lambda)\frac{d|S||\bar{S}|}{|V|},$$

where $\bar{S}$ is the complement of S and $\lambda \leq 0.9$ is the expansion parameter of G. It follows that the probability to stay in S after one step from a random vertex in S is

$$1 - \frac{|E(S, \bar{S})|}{d|S|} \leq 1 - (1 - \lambda)\frac{\bar{S}}{|V|} \leq 1 - (1 - \lambda)(1 - d\epsilon) \leq d\epsilon + \lambda$$

and since for any graph H, $\lambda(H^\ell) = (\lambda(H))^\ell$ it follows that the probability to end in S after a random walk of length ℓ starting in a random vertex of S is at most $d\epsilon + \lambda^\ell$.

We now have the facts needed to calculate a bound for $E[V^2]$. For j let I_j denote the random variable which is 1 if the j-th edge belongs to F and 0 otherwise. Then

$$\begin{aligned}
E[V^2] &\leq E[\left(\sum_j I_j\right)^2] = 2\sum_{\substack{j,j'\\ j\leq j'}} E[I_j \cdot I_{j'}] \\
&= 2\sum_j \left(\Pr[I_j = 1] \cdot \sum_{j'\geq j} \Pr[I_{j'} = 1 | I_j = 1]\right) \\
&\leq 2\sum_j \left(\Pr[I_j = 1] \cdot \sum_{j'\geq j} (\epsilon d + \lambda^{j'-j})\right) \\
&\leq 2\epsilon d \cdot \sum_j \left(\sum_{j'\geq j} \epsilon d + \sum_{k=0}^{\infty} \lambda^k\right) \\
&\leq 2\epsilon d\delta\sqrt{t}(\delta + 10) \\
&\leq 22\epsilon d\delta\sqrt{t}
\end{aligned}$$

In the second last inequality we used the assumption $\epsilon \leq \frac{1}{d\sqrt{t}}$ and in the last one we used $\delta < 1$. It now follows that

$$\Pr[V > 0] \geq \frac{E[V]^2}{E[V^2]} \geq \frac{(\frac{1}{2}\delta\sqrt{t}\epsilon)^2}{22\epsilon\delta\sqrt{t}d} = \frac{\sqrt{t}\epsilon}{3520d}.$$

□

To summarize, in this subsection we have proved the following.

Theorem 6.17. *There exists a real BSS algorithm which works in polynomial time that maps a nice QPS$(m, k, 2, s)$ instance ψ to a QPS$(d^{2t}m, 2\sqrt{t}k + (2\sqrt{t}+1)s, 2, d^{t+\sqrt{t}+1}s)$-instance ψ^t and has the following properties:*

- *If ψ is satisfiable, then ψ^t is satisfiable.*
- *If ψ is not satisfiable and $UNSAT(\psi) \leq \frac{1}{d\sqrt{t}}$, then $UNSAT(\psi^t) \geq \frac{\sqrt{t}}{3520d} \cdot UNSAT(\psi)$.*

Below we choose t such that $\frac{\sqrt{t}}{3520d} \gg 1$ in order to increase the gap sufficiently during an amplification step. This will compensate for a moderate decrease of the gap resulting from preprocessing and dimension reduction.

Note in particular that $UNSAT(\psi^t) \geq \frac{1}{d\sqrt{t}}$ if we apply amplification to a ψ that already satisfies $UNSAT(\psi) \geq \frac{1}{d\sqrt{t}}$.

The above construction works precisely the same in the complex BSS model.

6.2.4 Dimension reduction

The amplification step increases an unsatisfiability ratio $\epsilon < \frac{1}{d\sqrt{t}}$ by a factor $c \geq \frac{\sqrt{t}}{3520d}$. Thus starting with an unsatisfiable instance ϕ that has m constraints it would be sufficient to repeat the amplification step $\Omega(\log m)$ number of times in order to end with an instance that has a constant unsatisfiability gap $\geq \frac{1}{d\sqrt{t}}$. However, doing it naively the dimension of arrays in the resulting instance would no longer remain constant. This would imply as well the query complexity to be not any longer constant, compare Lemma 6.5. Therefore, the dimension has to be reduced again each time an amplification step was applied. This has to be done in such a way that we do not lose too much of the gap-increase the amplification step gave.

A first naive idea solving the task is to just change the way one looks upon the parameters of an instance. If every variable is considered as single array the new array size is 1. However, then the number of arrays on which constraints depend in the new instance is multiplied by the original array size. The problem is that the array size depends on the amplification factor. Then applying the amplification step another time we first have to preprocess again and the reduction of the gap in the preprocessing depends on the number of arrays on which a constraint depends. Altogether, the gap would be reduced instead of amplified.

To get around this problem one should first alter the instance in such a way that every constraint depends on at most Q variables only. Here, Q is an absolute constant independent of the array size. In Dinur's proof this is done using so-called transparent long proofs for NP. The corresponding construction is called alphabet-reduction there because the different amplification steps deal with satisfiability problems over finite alphabets of different cardinalities. With respect to the real number model it is more appropriate to consider it as a dimension reduction. All instances that occur during amplification are to be solved over the reals, i.e., there are no different 'alphabets' to deal with.

We already have used transparent long proofs in the construction of an algebraic proof for the $\mathrm{PCP}_{\mathbb{R}}$ theorem in the previous chapter. We saw that they are crucial for applying a technique called verifier-composition.

Here, long transparent proofs provide a way to replace each constraint in ψ^t by many constraints all depending on at most Q real variables (i.e., arrays of dimension 1); Q denotes the constant query complexity of the long transparent proof and thus is independent of the instance. If the old constraint is not satisfiable, then at least half of the new ones will not be satisfied. Thus one gets a reduction from constraints considered as QPS-instances to QPS-instances which blows up the gap to a constant. As seeming disadvantage the size of the long proof becomes double exponential in the size of the instance. But the verification using long proofs will be applied to instances of constant size only, namely single constraints in ψ^t. Thus the length of the transparent long proofs in fact does not matter at all. The much more important aspect is their structure which will be explained in detail below at the point where they are used for dimension-reduction.

The main result of this subsection is

Theorem 6.18. *There exists a BSS computable reduction which works in polynomial time and maps a QPS$(m(t), k(t), 2, s(t))$-instance ψ^t to a QPS$(\widehat{m}(t), C, Q, 1)$-instance $\widehat{\psi}^t$, where C, Q are constants, $\widehat{m}(t)$ is linear in $m(t)$ (the multiplication factor being double exponential in $s(t)$) and the following holds:*

- *If ψ^t is satisfiable, then so is $\widehat{\psi}^t$ and*
- *if ψ^t is unsatisfiable, then UNSAT$(\widehat{\psi}^t) \geq$ UNSAT$(\psi^t)/(160(d+1)^2)$.*

Proof. The general foregoing is as follows. First, we transform ψ^t into a slightly changed $\tilde{\psi}^t$ such that the array size (i.e., dimension) is doubled. All constraints in ψ^t become loops in the constraint graph of $\tilde{\psi}^t$; all non-loops in $\tilde{\psi}^t$ will be consistency constraints which claim the equality between certain halves of two of these double-sized arrays. After that, assignments for every double-sized array are replaced by a long transparent proof and loop-constraints are replaced by several proof checks. The consistency constraints are replaced by constraints checking the corresponding consistency requirement between the long transparent proofs which have replaced assignments of the arrays. This will give $\widehat{\psi}^t$. Now towards the details:

Claim 1: From instance ψ^t one can compute in polynomial time an instance $\tilde{\psi}^t \in$QPS$((d+1)m(t), \max\{k(t), s(t)\}, 2, 2 \cdot s(t))$ such that the following holds. If ψ^t is satisfiable so is $\tilde{\psi}^t$; otherwise, UNSAT$(\tilde{\psi}^t) \geq$ UNSAT$(\psi^t)/(80(d+1))$.

Proof of Claim 1: Consider the instance $\psi^t \in$ QPS$(m(t), k(t), 2, s(t))$ resulting from the amplification step. Denote its constraints by $C_1, \ldots, C_{m(t)}$. Change the instance by applying the construction of Lemma 6.9. The resulting instance $\tilde{\psi}^t$ belongs to QPS$((d+1)m(t), \max\{k(t), s(t)\}, 2, 2 \cdot s(t))$. Constraints in $\tilde{\psi}^t$ which arose from the old instance ψ^t we call pure constraints and denote them by $PC_1, \ldots, PC_{m(t)}$. The others are called consistency constraints and denoted by $CC_1, \ldots, CC_{dm(t)}$. A pure constraint depends on two old variable arrays and each such array occurs in precisely one pure constraint. Therefore, pure constraints group the old arrays in pairs. In the new instance $\tilde{\psi}^t$ each such pair is seen as own array of dimension $2 \cdot s(t)$. Consequently, the pure constraints are loops in the new constraint graph. The consistency constraints introduced by the construction of Lemma 6.9 claim equality between corresponding halves of the new variable arrays. According to Lemma 6.9 the gap is reduced by a factor of at most $80(d+1)$.

Next, our goal is to replace each pure and each consistency constraint by particular QPS-instances that depend on a constant number of variables only. This is done by means of long transparent proofs for $\mathrm{NP}_{\mathbb{R}}$, i.e., by a closer inspection of how these proof look like.

Notice: For all arguments below the following easy observation is crucial. Since s, t, d are constants the same holds for $s(t)$. Thus for all objects and running times depending superpolynomially on $s(t)$ these parameters still remain constant. The construction of long transparent proofs thus is only invoked for instances of constant size. This is also the reason why its structure is more important than the value of parameters for the long transparent verifier.

The construction being a bit different for the two types of constraints let us start with pure constraints $PC_i(u, v)$, where $u, v \in \mathbb{R}^{s(t)}$ are the two arrays the original constraint C_i in ψ^t depends on.

Claim 2: From PC_i we can construct an instance $QPC_i \in \mathrm{QPS}(2^{\exp(s(t))}, C, Q, 1)$ in polynomial time for suitable constants C, Q. Either both PC_i and QPC_i are satisfiable or at least half of the constraints in QPC_i are violated by any assignment.

Proof of Claim 2: The question whether an assignment for (u, v) satisfies PC_i is transformed to a $\mathrm{QPS}(\ell, 1, 3, 1)$-instance using folklore arguments in real number complexity theory [15]. More precisely, consider the evaluation of PC_i in a concrete assignment from $\mathbb{R}^{2s(t)}$ as computation of an algebraic circuit of size $poly(s(t))$ that is required to accept its input from $\mathbb{R}^{2s(t)}$. This QPS-instance depends on (u, v, z), where z represents all intermediate results of the circuit's computation and thus belongs to some $\mathbb{R}^{poly(s(t))}$. Note that as well ℓ is polynomially bounded in $s(t)$. Now we apply the construction of long transparent proofs for this $\mathrm{NP}_{\mathbb{R}}$ problem.

It shows that there exists a real number verifier which receives as input the following three strings:

- a representation of the $\mathrm{QPS}(\ell, 1, 3, 1)$-instance obtained from PC_i,
- a string of $\exp(\ell)$ random bits and
- a string π_i interpreted as verification proof for satisfiability of the QPS-instance. Here, π_i is of (algebraic) size $2^{\exp(s(t))}$.

In the first phase the verifier reads the random bits and makes a constant number Q of queries to the proof. For this part of the verification it needs (constant!) time $2^{poly(s(t))}$ to read the random bits. In the second phase the verifier reads the QPS instance and decides in polynomial time in $s(t)$ whether it accepts or rejects. The following conditions are satisfied:

- For a satisfiable $\mathrm{QPS}(\ell, 1, 3, 1)$-instance there exists a proof which the verifier will accept for every string of random bits;
- for an unsatisfiable $\mathrm{QPS}(\ell, 1, 3, 1)$-instance and for every proof the verifier rejects this proof for at least half of the random strings;
- the verifier queries a constant number Q of locations in the proof;
- in its decision phase the verifier checks whether these Q real numbers satisfy an instance in $\mathrm{QPS}(1, C, Q, 1)$ which it has calculated from the input instance and the random bits.

The last item above being the most important a few more words are necessary. The variables of all QPS-instances produced for different random strings are the coordinates of a given verification proof π_i. Since the verifier only reads Q of those the instances depend on Q real variables of dimension 1. For each fixed random string there is one constraint

generated by the verifier, its meaning being 'the verifier accepts π_i given the random string and the Q coordinates it inspects'. Finally, a closer look into Chapter 3 shows that all polynomials constituting this constraint have degree at most two and there are a finite number C of such polynomials. The number C results from the finitely many test-rounds that are performed by the verifier.[3]

There are $2^{\exp(s(t))}$ many random strings, so for every pure constraint PC_i we obtain a $\mathrm{QPS}(2^{\exp(s(t))}, C, Q, 1)$-instance QPC_i by building the union of all the constraints resulting from a fixed random string as described above. If PC_i is satisfiable so is QPC_i; and if PC_i is not satisfiable, a fraction of at least $\frac{1}{2}$ of the constraints in QPC_i is violated, no matter how a verification proof π_i looks like. Altogether, if $\tilde{\psi}^t$ is not satisfiable, then the minimum fraction of unsatisfied constraints in the union of all QPC_i is at least half of what it was in the original instance. This finishes the proof of Claim 2.

Next, we have to deal with the consistency constraints CC_j in $\tilde{\psi}^t$. Note that so far the variables on which different PC_i depend are treated as being different. So now the consistency requirements resulting from application of Lemma 6.9 have to be respected. This will be done by relating certain parts of the verification proofs $\pi_1, \ldots, \pi_{m(t)}$ appropriately. Again, such relations are expressed via suitable QPS-instances.

In order to understand the construction a closer look at the structure of long transparent verification proofs in Chapter 3 is needed. Suppose for two pure constraints $PC_1(u^{(1)}, v^{(1)})$ and $PC_2(u^{(2)}, v^{(2)})$ a consistency constraint CC_j in $\tilde{\psi}^t$ claims equality between, say, $u^{(1)}$ and $v^{(2)}$, both ranging over $\mathbb{R}^{s(t)}$. Let π_1, π_2 be candidates for long verification proofs for satisfiability of QPC_1 and QPC_2, respectively. Among other things π_1, π_2 are supposed to code two linear functions of arity $poly(s(t))$ which have as their coefficient-vectors satisfying assignments of the respective QPC-instances. However, since we are working over the reals these functions are supposed to be linear on a real set $\mathcal{X}_0 \subset \mathbb{R}^{poly(s(t))}$ only (and with respect to a finite set of scalar factors). Here, $\mathcal{X}_0$ is a finite set containing the standard basis of $\mathbb{R}^{polys(t)}$. The coding then is done by using a function value table that contains the values of the linear function on an appropriate superset $\mathcal{X}_1$ of $\mathcal{X}_0$. The cardinalities of $\mathcal{X}_1$ and $\mathcal{X}_0$ are of order $2^{\exp(s(t))}$ and $2^{poly(s(t))}$, respectively. Hence, the coding has double exponential size in $s(t)$ (which, however, is a constant). Taking into account the connection between assignments $(u^{(i)}, v^{(i)})$ for the original constraints PC_i and π_i this way of coding implies that both π_1 and π_2 in particular have to contain the following information: For each vector $r \in \mathcal{X}_0$ there is a coordinate in π_1 denoted by π_1^r and claiming a value for $r^T \cdot u^{(1)}$. This is true because the encoding π_1 in particular contains as part an encoding of the linear function from $\mathcal{X}_0 \to \mathbb{R}$ given by $u^{(1)}$

[3] If the reader does not want to dive into the construction of real long transparent proofs one can alternatively argue here as follows: The verifier computes its result in $poly(s(t))$ many steps once the Q coordinates have been determined. Describe this deterministic computation once again as above in terms of a $\mathrm{QPS}(1, poly(s(t)), \tilde{Q}, 1)$-instance, where all polynomials involved in the new constraint are of degree at most 2 and depend on at most 3 variables. Here, $\tilde{Q}$ depends polynomially on Q and $s(t)$. Since we start from a fixed s and since t is constant we obtain the desired statement, but with different constants for C and Q.

as coefficient vector. The above coordinate just contains the value of this function in the argument r. Similarly, there is a coordinate in π_2 claiming a value for $r^T \cdot v^{(2)}$.

If the verifier accepts with large enough probability then from the verification proof π_1 related to PC_1 a unique function $g_1 : \mathcal{X}_0 \mapsto \mathbb{R}$ can be defined such that the following holds:[4]

- g_1 is linear on $\mathcal{X}_0$ (with respect to a suitable set of scalar factors),
- the fraction of $r \in \mathcal{X}_0$ such that $g_1(r) \neq \pi_1^r$, i.e., $g_1(r)$ disagrees with the value π_1 claims for $r^T \cdot u^{(1)}$, is at most $\frac{1}{8}$ and
- any other linear function on $\mathcal{X}_0$ differs from g_1 in at least half of its values.

A function g_2 having analogous properties with respect to $v^{(2)}$ can be defined from the verification proof π_2 of the pure constraint PC_2 if it is accepted with high enough probability.[5]

Now in order to replace the consistency constraint $CC_j : u^{(1)} = v^{(2)}$ we add for all $r \in \mathcal{X}_0$ as own constraint the equation $\pi_1^r = \pi_2^r$ to the new instance $\widehat{\psi}^t$. This equation is of degree 1 since it relates two coordinates of π_1, π_2 and these coordinates represent (part of) the variables of $\widehat{\psi}^t$ as explained further up. Since $\mathcal{X}_0$ is of cardinality $2^{poly(s(t))}$ constraint CC_j is thus replaced by a QPS$(2^{poly(s(t))}, 1, 2, 1)$-instance which we denote by QCC_j.

Without loss of generality we finally assume that the number of constraints added to the new instance $\widehat{\psi}^t$ for any constraint in the old instance $\tilde{\psi}^t$ is the same. If this is not yet the case just copy this set of constraints a number of times until the assumption is satisfied. This is easily achieved since all QPC_i have the same number of constraints and the same holds for the QCC_j. This ends the construction of $\widehat{\psi}^t$. It consists of the union of all constraints in one of the QPC_i or one of the QCC_j. Its variables are the coordinates of verification proofs $\pi_1, \ldots, \pi_{m(t)}$.

It finally remains to show

Claim 3: The satisfiability gap of $\widehat{\psi}^t$ is at most by a constant factor of $\frac{1}{160(d+1)^2}$ smaller than that of ψ^t.

Proof of Claim 3: Consider any assignment $(\pi_1, \ldots, \pi_{m(t)})$ to the variables of $\widehat{\psi}^t$ and assume a fraction of at least $\frac{1}{80(d+1)}\epsilon$ constraints in $\tilde{\psi}^t$ to be violated by any assignment to its variables. Define such an assignment resulting from $(\pi_1, \ldots, \pi_{m(t)})$ as follows: Consider a pure constraint PC_i together with its corresponding new instance QPC_i and π_i. If the part of the verifier that checks π_i accepts with large enough probability, then as explained above π_i can be used to define an assignment $(u^{(i)}, v^{(i)})$ for the variables of PC_i by setting $u_k^{(i)} := g_1(e_k), 1 \leq k \leq s(t)$, where g_1 is the corresponding linear function resulting from

[4] These linear functions correspond to the Walsh-Hadamard code in the discrete framework.

[5] Note a technical detail here: The verifier guarantees the function value table to be close to a linear function on $\mathcal{X}_0$. It is easy to extend this verifier without using significantly more resources in such a way that this property also holds for subtables representing functions of a smaller arity. This is tacitly used above for coding the linear functions with coefficient vectors $u^{(1)}$ and $v^{(2)}$, respectively.

the first $s(t)$ variables in π_i and e_k is the k-th unit vector in $\mathbb{R}^{s(t)}$; similarly for $v^{(i)}$. Note that this assignment satisfies the pure constraint PC_i due to the properties of the verifier. If the verifier does not accept π_i with large enough probability (and thus there might not exist such a g_1^+) we define $u_k^{(i)}, 1 \leq k \leq s(t)$ just as $\pi_i^{e_k}$, i.e., the coordinate in π_i where ideally in a correct proof the value of the corresponding linear function coded by π_i in point e_k would be expected to be represented. That way an assignment $(u^{(1)}, \ldots, v^{m(t)})$ for $\tilde{\psi}^t$ is obtained.

As already argued above for each pure constraint PC_i that is violated by this assignment at least half of the corresponding constraints in QPC_i are violated by π_i. For a violated consistency constraint CC_j involving again, say, PC_1, PC_2 and violating $u^{(1)} = v^{(2)}$ it follows that at least half of the constraints in QCC_j are violated, at least half of the constraints in QPC_1 are violated or at least half of the constraints in QPC_2 are violated. This is the case because $u^{(1)}$ and $v^{(2)}$ generate different linear functions and because two linear functions on $\mathcal{X}_0$ differ on at least half of their arguments.

To conclude we have shown that both for the sets of violated pure and violated consistency constraints the gap is reduced by at most a factor $2(d+1)$ since every pure constraint is connected to at most d consistency constraints. Since Lemma 6.9 reduced it by another factor of at most $\frac{1}{80(d+1)}$ altogether the reduction factor is at most $\frac{1}{160(d+1)^2}$. Claim 3 follows, and this finishes the proof of the theorem. □

6.2.5 Putting all together

We start with an instance ϕ of the $\mathrm{NP}_{\mathbb{R}}$-complete problem $\mathrm{QPS}(m, 1, 3, 1)$ and consider it as instance in $\mathrm{QPS}(m, C, Q, 1)$.[6] Here, $Q \geq 3$ is the $O(1)$-constant from long transparent proofs for $\mathrm{QPS}(m, 1, 3, 1)$ and $C \geq 1$ is the number of polynomials in a proof check as explained in the proof of Theorem 6.18, i.e., the number of polynomials in the QPS-instance which the verifier computes out of the input instance and the random bits. Therefore, both C and Q are constant.

Now applying preprocessing, amplification and dimension reduction with a suitable value for t leads to a $\mathrm{QPS}(m', C, Q, 1)$-instance $\widehat{\psi}^t$ such that

- m' is linear in m, the multiplication factor is double exponential in Qd^t;
- if ϕ is satisfiable so is $\widehat{\psi}^t$;
- if ϕ is unsatisfiable and $\mathrm{UNSAT}(\phi) \leq \frac{1}{d\sqrt{t}}$, then $\mathrm{UNSAT}(\widehat{\psi}^t) \geq \mathrm{UNSAT}(\phi) \cdot \frac{\sqrt{t}}{10^{10}Qd^4}$.

Assume ϕ is not satisfiable. We now choose $t = (2 \cdot 10^{10}Qd^4)^2$ so that a gap which is smaller than $\frac{1}{d\sqrt{t}}$ will be amplified with a factor of at least 2 by this reduction. If we repeat this process as long as the gap is smaller than $\frac{1}{d\sqrt{t}}$ we will obtain an instance with a gap of at least $\frac{1}{d\sqrt{t}}$ within at most $\log m$ steps.

[6] This viewpoint is taken because preprocessing, amplification and dimension-reduction are repeated several times.

Finally, since in every step the number of constraints increases linearly, after less than $\log m$ steps the number of constraints in the final instance is polynomial in m. Using Lemma 6.5 we thus arrive at the Main Theorem:

Theorem 6.1. *(PCP-Theorem over $\mathbb{R}$ and $\mathbb{C}$) In the real BSS model it is*

$$\mathrm{NP}_{\mathbb{R}} \;=\; \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1)).$$

The analogue statement is true for the BSS model over the complex numbers.

Note finally that neither the construction of long transparent proofs as given in Chapter 3 nor the arguments given in this chapter depend on the order available in the real number model. Therefore, all arguments remain true when performed in the complex BSS model.

Chapter 7

Conclusion

We showed that the PCP theorem holds as well in the BSS model over $\mathbb{R}$ and over $\mathbb{C}$. This adds another important theorem to the list of results that hold in the BSS framework as well. By giving two different proofs of the $\text{PCP}_{\mathbb{R}}$ theorem (one along the lines of Arora et al. and one along the lines of Dinur) we obtain insight into the usability of classical proof techniques in the BSS model of real and complex number computation.

Though one may expect that in general algebraic classical proof techniques might be useful in the BSS framework as well whereas classical combinatorial techniques probably do not work for BSS machines, the opposite seems to be true in this case.

The proof by Dinur uses expander graph theory and works a lot with manipulations of finite alphabets. BSS machines over $\mathbb{R}$ and $\mathbb{C}$ always work with infinite alphabets. However it turns out that one can find a translation in which the alphabet size corresponds to the dimension of a real or complex vector. Using this translation most of the classical proof goes through in the BSS model as well.

The proof by Arora et al. relies a lot on properties of finite fields. Even though all the algebraic techniques used in the proof can in principle be used in the BSS model as well, it seems impossible to obtain the same strong results because $\mathbb{R}$ is not finite. A lot of new ideas are necessary to make things work. The main idea that seems necessary is to use trigonometric polynomials to connect properties of finite fields with the infinity of $\mathbb{R}$. Still, some of the main partial results (e.g. the low-degree test and the segmentation procedure) are weaker and have longer proofs than their classical counterparts.

There are many questions which could be addressed as a next step. One obvious question would be to reduce the number of queries that a PCP-verifier makes. This number is a constant and one may ask how small this constant can be for a PCP-verifier which decides an NP-complete problem. One possible approach to this question could be to try to prove a BSS equivalent of the following result by Håstad [29].

Theorem 7.1. *For every $\delta > 0$ and ever language $L \in \text{NP}$, there is a PCP-verifier V for L making three queries. Every input $x \in L$ is accepted with probability $\geq 1 - \delta$ and every input $x \notin L$ accepted with probability $\leq \frac{1}{2} + \delta$.*

In the proof of this theorem Håstad makes use of Fourier expansions. Perhaps trigonometric polynomials might again be useful in proving an equivalent theorem in the BSS model.

Another general question which might be interesting is whether trigonometric polynomials could be used for other purposes as well in BSS theory. As far as we know there has not been any research done on this topic yet. Given their huge importance in many different areas this may be something worth looking at. Though trigonometric polynomials probably are not so relevant for classes such as $\mathrm{P}_{\mathbb{R}}$ because a BSS machine cannot compute the sine function, for classes such as $\mathrm{NP}_{\mathbb{R}}$ they might be useful because a BSS machine can verify the value of $(\cos x, \sin x)$ for $x \in \mathbb{Q}$.

The PCP theorem in the Turing model turned out to be extremely useful in obtaining non-approximability results for many problems. It was for example used to establish that the problem of finding the maximum number of simultaneously satisfiable clauses in a 3SAT formula does not have a polynomial time approximation scheme. For 3SAT and many other approximation problems it often could be shown that the known approximation algorithms are optimal (unless P = NP). In the BSS model approximability problems have not been studied that much. This theory could be developed and compared with the theory of approximability problems in the Turing model. One could for example try to approximate the QPS problem.

Question 7.2. *Given $\epsilon > 0$. Does there exist a polynomial time BSS algorithm which meets the following requirement. For each QPS instance where $\alpha \leq 1$ is the largest fraction of polynomials which have a common zero the algorithm outputs a number between α and $\frac{\alpha}{1+\epsilon}$.*

If for every $\epsilon > 0$ such an algorithm would exist, then by using the gap reduction from Chapter 6 one could construct a polynomial time algorithm to decide the QPS problem. This would imply $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$. So unless $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$ there does not exist a polynomial time approximation scheme for QPS. At first sight it seems even unclear whether there exists an $\epsilon > 0$ for which such a polynomial time algorithm exists. This may be another interesting question to look at. In this text we use the QPS problem as a BSS-analog of the 3SAT problem and the MAX-3SAT problem clearly has a constant factor approximation algorithm. If this would not be the case for the MAX-QPS problem, then that would be a fundamental difference.

One could also look for other approximation problems in the BSS context and see if the $\mathrm{PCP}_{\mathbb{R}}$ theorem can be applied to establish inapproximability results as has been done successfully in the Turing model. Given that approximation theory plays an important role in the Turing model it could be worthwile to see if a similar kind of theory could be developed in the BSS model.

The low-degree test that we use in our algebraic proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem uses trigonometric polynomials because the existing low-degree tests for algebraic polynomials have unsuitable structure for our purposes. Also, we were sceptical that we could design a low-degree test for algebraic polymials with a suitable structure. That is why we used

trigonometric polynomials. However, using the segmentation procedure from Chapter 5 (which relies on the low-degree test for trigonometric polynomials) it becomes possible to construct a low-degree test for algebraic polynomials such that only a constant number of small segments need to be queried by the test procedure. Using this kind of test in the proof of the $\mathrm{PCP}_{\mathbb{R}}$ theorem would only complicate things. Though this test would be relatively complicated compared to other low-degree tests for algebraic polynomials over $\mathbb{R}$, it might be worthwile to construct this test because in this test the verifier only has to query a constant nubmer of small segments whereas in the other tests the number of segments queried is logarithmic in the number of variables. Perhaps this could be useful in other kinds of proof verification settings.

Another kind of proof verification that can also be considered in the BSS model are interactive proofs. In the Turing model the class of languages for which an interactive verification protocol exists is called IP. This class equals PSPACE, see [42]. A complexity class $\mathrm{IP}_{\mathbb{R}}$ can naturally be defined in the BSS model. The situation for PSPACE is a bit different because space restrictions alone are not relevant in the BSS model because with a simple coding trick everything can be computed in linear space. Therefore the problems in the class $\mathrm{PSPACE}_{\mathbb{R}}$ need not only be computed within polynomial space but also within exponential time. In the Turing model the class PSPACE equals the class of problems that can be computed in parallel polynomial time and also equals the class of problems that can be computed in alternating polynomial time. The corresponding complexity classes in the BSS model are all different. So maybe $\mathrm{IP}_{\mathbb{R}}$ equals one of them, but it could also be that $\mathrm{IP}_{\mathbb{R}}$ is the fourth in a set of different complexity classes who's classical counterparts are all the same.

Some results in this direction have been obtained in [33] and in [8]. The larger part of the first paper deals with interactive protocols for additive BSS machines. One interesting result which is obtained there for the class $\mathrm{IP}_{\mathbb{R}}$ (for multiplicative BSS machines) is that this class is not contained in the class of problems computable by a BSS machine in parallel polynomial time. So $\mathrm{IP}_{\mathbb{R}}$ contains problems with high computational complexity. On the other hand, only the trivial lower bound $\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{IP}_{\mathbb{R}}$ is known. It is not even clear whether co-$\mathrm{NP}_{\mathbb{R}}$ is contained in $\mathrm{IP}_{\mathbb{R}}$. In the second paper it is proved that $\mathrm{IP}_{\mathbb{R}}$ is contained in $\mathrm{MA}\exists_{\mathbb{R}}$. Roughly speaking this is the class of problems which allows a polynomial number of quantifications whereby quantifications over $\{0,1\}$ may be existential and universal, but quantifications over $\mathbb{R}$ may only be existential.

A first idea for obtaining a non-trivial lower bound for $\mathrm{IP}_{\mathbb{R}}$ may be to try to adapt the techniques used by Shamir to show that IP = PSPACE to the BSS model. However, at first sight it seems unclear how to do this. In the proof that IP = PSPACE a technique is used which transforms a quantified boolean formula into an algebraic polynomial. Hereby existential and universal quantifiers are replaced by sums and products. In the case that quantifiers range over $\mathbb{R}$ instead of over $\{0,1\}$ it seems unclear how to do this.

Bibliography

[1] S. Arora, B. Barak: Computational Complexity: A Modern Approach. Cambridge University Press, 2009.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy: Proof verification and hardness of approximation problems. Journal of the ACM Vol. 45 (3), 501–555, 1998. Preliminary version: Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, 14–23, 1992.

[3] S. Arora, S. Safra: Probabilistic checking proofs: A new characterization of NP. Journal of the ACM Vol. 45 (1), 70–122, 1998. Preliminary version: Proc. 33rd Annual IEEE Symposium on the Foundations of Computer Science, 2–13, 1992.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer, 1999.

[5] M. Baartse, K. Meer: The PCP theorem for NP over the reals. Foundations of Computational Mathematics Vol. 15 (3), Springer, 651-680, 2015.

[6] M. Baartse, K. Meer: Topics in real and complex number complexity theory. Recent Advances in Real Complexity and Computation, Contemporary Mathematics Vol. 604, J.L. Montana, L.M. Pardo (Eds.), American Mathematical Society, 1–53, 2013.

[7] M. Baartse, K. Meer: Testing low degree trigonometric polynomials. Proc. 9th Computer Science in Russia, Springer Lecture Notes in Computer Science Vol. 7913, E.A. Hirsch, S.O. Kuznetsov, J. Pin, N.K. Vereshchagin (Eds.), 77–96, 2014.

[8] M. Baartse, K. Meer: Some results on interactive proofs for real computations. To appear in Proc. 11th Conference on Computability in Europe.

[9] L. Babai: E-mail and the unexpected power of interaction. Proc. 5th Annual Structure in Complexity Theory Conference, IEEE, 31–91, 1990.

[10] L. Babai: Transparent proofs and limits to approximation. First European Congress of Mathematics, Progress in Mathematics Vol. 3, A. Joseph, F. Mignot, F. Murat, B. Prum, R. Rentschler (Eds.), Birkhäuser Basel, 31–91, 1994.

[11] S. Basu, R. Pollack, M.F. Roy: On the combinatorial and algebraic complexity of quantifier elimination. Journal of the ACM Vol. 43 (6), 1002–1045, 1996.

[12] S. Basu, T. Zell: Polynomial hierarchy, Betti numbers, and a real analogue of Toda's theorem. Foundations of Computational Mathematics Vol. 10 (4), Springer, 429–454, 2010.

[13] C. Beltrán, L.M. Pardo: Efficient polynomial system-solving by numerical methods. Journal of Fixed Point Theory and Applications Vol. 6, Springer, 65–85, 2009.

[14] L. Blum, F. Cucker, M. Shub, S. Smale: Complexity and Real Computation. Springer, 1998.

[15] L. Blum, M. Shub, S. Smale: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bulletin of the AMS Vol. 21, 1–46, 1989.

[16] O. Bournez, M.L. Campagnolo: A Survey on Continuous Time Computations. New Computational Paradigms. Changing Conceptions of What is Computable. S.B. Cooper, B. Löwe, A. Sorbi (Eds.), Springer, 383–423. 2008.

[17] P. Bürgisser, M. Clausen, M.A. Shokrollahi: Algebraic Complexity Theory Vol. 315 of *Grundlehren*. Springer, 1997.

[18] P. Bürgisser, F. Cucker: On a Problem Posed by Steve Smale. Annals of Mathematics Vol. 174 (3), Princeton University and the Institute for Advanced Study, 1785–1836, 2011.

[19] O. Chapuis, P. Koiran: Saturation and stability in the theory of computation over the reals. Annals of Pure and Applied Logic Vol. 99, Elsevier, 1–49, 1999.

[20] G.E. Collins: Quantifier Elimination for real closed fields by Cylindrical Algebraic Decomposition. Springer Lecture Notes in Computer Science Vol. 33, 134–183, 1977.

[21] I. Dinur: The PCP theorem by gap amplification. Journal of the ACM Vol. 54 (3), 2007.

[22] R. O'Donnell: A history of the PCP theorem. Unpublished note; downloaded on January 28, 2015 from `http://courses.cs.washington.edu/courses/cse533/05au/`

[23] N. Fitchas, A. Galligo, J. Morgenstern: Precise sequential and parallel complexity bound for quantifier elimination over algebraically closed fields. Journal of Pure and Applied Algebra Vol. 67, Elsevier, 1–14, 1990.

[24] O. Goldreich: Basic Facts about Expander Graphs. Studies in Complexity and Cryptography, Springer Lecture Notes in Computer Science Vol. 6650, O. Goldreich (Ed.), 451–464, 2011.

[25] H. Fournier, P. Koiran: Are lower bounds easier over the reals? Proc. 30th Annual ACM Symposium on Theory of Computing, J.S. Vitter (Ed.), 507–513, 1998.

[26] H. Fournier, P. Koiran: Lower Bounds Are not easier over the reals: Inside PH. 27th Proc. International Colloquium Automata, Languages and Programming, Springer Lecture Notes in Computer Science Vol. 1853, U. Montanari, J.D. Rolim, E. Welzl (Eds.), 832–843, 2000.

[27] P. Frisco: Computing with Cells - Advances in Membrane Computing. Oxford University Press, 2009.

[28] D.Y. Grigoriev: Complexity of Deciding Tarski Algebra. Journal of Symbolic Computation Vol. 5, Elsevier, 65–108, 1988.

[29] J. Håstad: Some optimal inapproximability results. Journal of the ACM Vol. 48 (4), 798–859, 2001.

[30] S. Haykin: Neural Networks - A Comprehensive Foundation. Prentice Hall, 2nd edition, 1999.

[31] J. Heintz, M.F. Roy, P. Solerno: On the complexity of semialgebraic sets. Proc. International Federation for Information Processing 11th World Computer Congress, G.X. Ritter (Ed.), Elsevier, 293–298, 1989.

[32] D. Hougardy, H.J. Prömel, A. Steger: Probabilistically checkable proofs and their consequences for approximation algorithms. Discrete Mathematics Vol. 136, Elsevier, 175–223, 1994.

[33] S. Ivanov, M. de Rougemont: Interactive Protocols on the reals. Computational Complexity Vol. 8, Springer, 330–345, 1999.

[34] C. Lund, L. Fortnow, H. Karloff, N. Nisan: Algebraic methods for interactive proof systems. Journal of the ACM Vol. 39 (4), 859–868, 1992.

[35] K. Meer: Transparent long proofs: A first PCP theorem for $\mathrm{NP}_{\mathbb{R}}$. Foundations of Computational Mathematics Vol. 5 (3), Springer, 231–255, 2005.

[36] K. Meer: Almost transparent short proofs for $\mathrm{NP}_{\mathbb{R}}$. Proc. 18th International Symposium on Fundamentals of Computation Theory, Springer Lecture Notes in Computer Science Vol. 6914, 41–52, 2011.

[37] K. Meer, M. Ziegler: An explicit solution to Post's problem over the reals. Journal of Complexity Vol. 24 (1), Elsevier, 3–15, 2008.

[38] M.A. Nielsen, I.L. Chuang: Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[39] O. Reingold, S. Vadhan, A. Wigderson: Entropy waves, the zig-zag graph product, and new constant degree expanders. Annals of Mathematics Vol. 155, Princeton University and the Institute for Advanced Study, 157–187, 2002.

[40] J. Renegar: On the computational Complexity and Geometry of the first-order Theory of the Reals , I - III. Journal of Symbolic Computation Vol. 13, Elsevier 255–352, 1992.

[41] R. Rubinfeld, M. Sudan: Self-testing polynomial functions efficiently and over rational domains. Proc. 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, G.N. Frederickson (Ed.), 23–32, 1992.

[42] A. Shamir: IP = PSPACE. Journal of the ACM Vol. 39 (4), 869–877, 1992.

[43] A. Tarski: A decision method for elementary algebra and geometry. University of California Press, 2nd edition, 1951.

[44] K. Weihrauch: Computable Analysis: An Introduction. Springer, 2000.

[45] H. Woźniakowski: Why does information-based complexity use the real number model? Theoretical Computer Science Vol. 219 (1-2), Elsevier, 451–465, 1999.

[46] R. Zippel: Probabilistic algorithms for sparse polynomials. Proc. International Symposium on Symbolic and Algebraic Computation, Springer Lecture Notes in Computer Science Vol. 72, 216–226, 1979.